Syntaxe, raisonnement et génomes

Jacques Nicolas

HABILITATION À DIRIGER DES RECHERCHES

présentée devant

L'Université de Rennes 1
Spécialité : Informatique

par

Jacques NICOLAS

SYNTAXE, RAISONNEMENT ET GENOMES

soutenue le 13 Mai 2008 devant le jury composé de :

Prof.	Olivier Ridoux	Président
Prof.	Alain Denise	Rapporteurs
Prof.	Jacques Cohen	
Prof.	Maxime Crochemore	
Prof. Dr.	Torsten Schaub	Examinateurs
Dr	Antoine Danchin	

Remerciements

Je remercie Olivier Ridoux, directeur de l'IFSIC et digne représentant de la créativité et de la curiosité universitaire, qui me fait l'honneur de présider le jury.

Je remercie également les rapporteurs du jury de me faire le très grand honneur d'accepter cette contrainte. Ils ont tous abordés le virage de la bioinformatique avec des itinéraires différents, mais avec un degré d'investissement commun et la volonté de transmettre la beauté des problèmes associés en enseignement. Alain Denise est un homme chaleureux et compétent qui fait partie du cercle restreint qui m'a convaincu de produire ce document. Maxime Crochemore marie avec bonheur une culture et une profondeur d'investigation qui m'étonne encore sur l'algorithmique des textes. Jacques Cohen est pour moi le parangon du chercheur universitaire, qui fait aimer ce métier à toute personne qui a eu la chance de le croiser.

Antoine Danchin a été un passeur dans ma vie de chercheur. J'ai découvert avec lui la beauté des structures moléculaires du vivant et je n'ai pas regretté depuis lors une seule seconde cet appel scientifique aux communautés en mathématique, physique et informatique qu'il avait organisé alors à Gif sur Yvette avec d'autres personnes comme J.-L. Risler et A. Hénaut. Qu'il en soit infiniment remercié. Je lui souhaite de tout coeur de trouver un mécène pour continuer son exploration passionnante et passionnée des origines de la vie au delà du temps court imposé par le système de retraite...

Bien sûr, la présence de Torsten Schaub ne doit rien au hasard. C'est un professionnel compétent, une personnalité accueillante et qui sait ne jamais se prendre trop au sérieux, et quelqu'un avec qui j'aurai un immense plaisir à collaborer dans les années à venir dans ce champ immense de la modélisation de connaissances et du raisonnement automatique.

Merci enfin à toute l'équipe Symbiose qui est une source de joies scientifiques et humaines et bravo à ces générations de thésards qui ont supporté stoïquement mes gribouillis isotropes et noircisseurs que j'ai dispensés sans la moindre autorisation. Mon expérience m'autorise un seul conseil : le chemin exigeant de la compréhension fuit les certitudes sereinement, voire joyeusement ! Une dédicace particulière à la garde rapprochée des fidèles, Anne, Catherine, Dominique, François, Olivier, Rumen, Véronique, relecteurs de la dernière heure...

Et puis Laurent Trilling, mon maître, qui m'a largement laissé la bride sur le cou pendant ma thèse, tout en me formant à l'essentiel, la beauté informatique (dérepérage, funarg, Prolog, et tant d'autres ...) et l'art de poser des questions naïves. Laurent, je te dédie ce document.

à Sempé et Doisneau

Ce texte est dédié aux personnes pour qui la recherche, comme toute activité de création, se traduit par une démarche globale d'accueil, d'échange et d'humilité.

"We are perishing for want of wonder, not for want of wonders"
(Nous périssons faute d'émerveillement, et non pas faute de merveilles.)
G.K. Chesterton

Ce qui est plus décisif que le regard scientifique,
c'est la capacité à écouter les autres et à s'expliquer avec eux,
c'est-à-dire à établir la communication.
Quand il y a interaction, les éléments réussissent à faire apparaître quelque chose
qui n'était ni dans l'un ni dans l'autre et qui apparaît parce qu'ils sont ensemble.
Pour les Chrétiens, c'est la fameuse phrase :
" Lorsque vous serez réunis, je serai parmi vous " .
C'est aussi une phrase de scientifique :
Lorsque vous êtes en état d'intercommunication,
un plus apparaît et ce plus ne peut apparaître que s'il y a capacité à communiquer.
Grâce à l'invention du langage, sous toutes ses formes,
nous avons inventé un ensemble plus riche que chacun d'entre nous.
Albert Jacquard. 8 sept. 1998

Table des matières

Chapitre 1

Quelques défis de la bioinformatique

Une seule question peut être plus explosive que mille réponses.
Jostein Gaarder, Le monde de Sophie

J'ai des questions à toutes vos réponses.
Woody Allen

La bioinformatique est un domaine extrêmement dynamique, qui se laisse difficilement circonscrire par une définition trop restrictive. Contentons-nous de la caractériser dans ce cadre comme un ensemble d'études de nature mathématique et informatique visant à assister le biologiste dans sa compréhension du vivant au niveau moléculaire, et ceci face à l'arrivée récente et en masse de données d'observations.

Il ne s'agit pas pour moi dans ce petit document de dresser une énième liste des problèmes de nature informatique que posent les avancées technologiques révolutionnaires de ces dernières années en biologie. Au-delà des besoins classiques de gestion de ces données pléthoriques, il existe en effet un vaste champ d'application de méthodes et d'outils informatiques ainsi que de formalisation de nouveaux problèmes informatiques associés à de nouveaux problèmes biologiques. D'autres auteurs ont écrit avec pertinence sur le sujet et je ne peux que conseiller la lecture de livres comme [99, 172] ou de certaines synthèses comme [43] pour se faire une idée de la variété des thèmes abordés en bioinformatique.

Je souhaite au contraire commencer par ce que les auteurs réservent en général à la conclusion : offrir un point de vue personnel et pas forcément majoritaire des développements scientifiques nécessaires, pour attaquer les défis de cette nouvelle discipline que constitue la compréhension à la fois fine et globale des mécanismes du vivant.

Tout d'abord, il faut s'interroger sur les spécificités du vivant par rapport à d'autres domaines comme la physique ou la chimie qui ont déjà pleinement intégré leur mutation mathématique et informatique. Ce positionnement fait peur encore parfois et fait débat car il s'agit d'un problème

existentiel pour les biologistes. Bien sûr, la biologie est soumise aux lois de la physique et de la chimie et les résultats obtenus doivent être compatibles avec ses lois. Cependant, la complexité ou simplement la structure des systèmes à étudier n'a rien à voir avec ce qui est habituellement traité en physique. Si biomathématique, biophysique et biochimie ne sont pas des disciplines nouvelles, l'arsenal scientifique et méthodologique nécessaire pour comprendre ou même simplement définir les propriétés d'un système biologique est encore balbutiant.

Des auteurs comme Antoine Danchin [56], Gilbert Chauvet [39], ou Évelyne Fox-Keller [82] par exemple ont su très bien décrire quelques éléments fondamentaux de ces systèmes :

- **L'hétérogénéité**. Un système biologique est construit à partir de très nombreux types de composants différents, chaque composant étant présent généralement en petit nombre dans un espace réduit.
- **La hiérarchisation**. Un système biologique se structure à l'aide de membranes qui isolent des sous-systèmes à différents niveaux d'échelle. Ces emboîtements sont associés à un contrôle du passage de matière d'un niveau à l'autre.
- **L'interaction**. Tous les composants sont en interaction[1], ce qui mène à une combinatoire globale très élevée de l'assemblage de ces relations pour aboutir au déterminisme fonctionnel de l'ensemble. L'interaction peut-être spatiale (locale) ou par message avec un adressage pouvant traverser plusieurs couches.
- **La mémorisation**. Un système biologique évolue tout en conservant une trace de ses évolutions[2]. C'est un mécanisme inscrit dans les quatre dimensions (spatio-temporel), qui implique une dynamique non réversible qui dépend potentiellement de toute l'histoire du système.
- **La robustesse**. Un système biologique, comme tout système complexe, tend à posséder une certaine inertie face à des petites variations de l'environnement. Cet aspect est renforcé en biologie par le mécanisme des membranes et par la conservation de sous-systèmes fonctionnellement redondants. Plus les systèmes vivants se complexifient et plus ils s'isolent des effets de l'environnement (ce qu'on appelle l'homéostasie en physiologie). Ce dernier aspect rend réaliste les tentatives intégratives de la biologie car on peut espérer réduire le modèle de fonctionnement du système malgré les nombreux événements non maîtrisables susceptibles d'agir sur celui-ci.

La majorité des théoriciens de la biologie recherche des modèles mathématiques ayant au minimum une capacité prédictive et dans l'idéal une valeur explicative. La place possible de l'informatique dans ses constructions est en général sous-estimée. On pense à l'instrument scientifique et pas à la discipline scientifique : construction de bases de données, partage de connaissances et d'outils d'analyse de données sur le Web, accélération de calculs pour la simulation, rien qui soit particulier au monde de la biologie.

Il nous semble cependant que l'informatique en tant que science est pleinement concernée par ce domaine pour au moins deux raisons :

[1] Antoine Danchin a choisi la métaphore de la barque de Delphes, composée de planches individuellement interchangeables, pour souligner le caractère fondamental de ces relations par rapport aux composants.

[2] Beaucoup de biologistes reprennent à leur compte la devise de T. Dobzhansky : « Nothing in biology makes sense except in the light of evolution [62]. »

– **Les systèmes vivants sont des machines symboliques**. Ils utilisent un code. Ils élaborent de l'information discrète à partir de signaux continus. Ils véhiculent et stockent des messages.
  Au niveau des traces statiques, les macromolécules du vivant sont des supports clés pour cela. Il est donc normal qu'un grand nombre d'apports mais aussi de questions informatiques émargent à l'algorithmique des séquences, mais également à celle des graphes de par l'importance des relations qu'elles entretiennent entre elles.
  Au niveau dynamique, de nombreuses molécules interviennent comme autant de variables d'état qui vont influer sur les transitions futures du système, mais qui concourent, du fait de la robustesse, à des comportements essentiellement qualitatifs. Enfin, on peut même parler de sémantique[3] au sens où chaque fonction est le résultat d'une adaptation à l'environnement.
  Langages, machines, c'est un champ de modélisation pour l'informatique mais c'est également un nouvel espace de machines à explorer.
– **Il existe des échanges permanents entre un niveau informationnel (le niveau du génome) et un niveau biochimique (le niveau des interactions internes ou externes)**. La logique de fonctionnement des systèmes vivants est de ce fait particulièrement complexe et expliquer ce fonctionnement à différents niveaux d'abstraction est un processus hors de portée de simples systèmes d'équations dans la continuité des lois construites en physique. Il faut y ajouter un niveau algorithmique.
  La biologie est une science expérimentale par excellence qui est elle-même un champ d'expérimentation fantastique pour la formalisation des processus d'acquisition de connaissances. J'écris bien connaissances, c'est-à-dire un ensemble cohérent, structuré, argumentable de faits et non pas simplement des données. La biologie vit non seulement un changement d'échelle mais également un changement de méthodes : de même que le recueil des données a dû être fortement automatisé, il n'est plus possible à la main de produire des hypothèses, puis vérifier et intégrer les résultats.
  Algorithmique, logique, raisonnement automatique, assistance à la compréhension des phénomènes complexes, nous sommes là encore au coeur de la discipline informatique.

Pour une vue plus complète des enjeux et des défis, la littérature est abondante (e. g. [34, 165, 33]). Je ne saurais trop conseiller la lecture du rapport bien équilibré de la National Academy of Science américaine écrit en 2005 à la demande de la NSF, Le DoE, le NIH et le DoD [163]. Nous sommes particulièrement en phase avec le fait que c'est autour des problèmes biologiques que doit se construire l'interdisciplinarité et soutenons fortement pour cela la mise en place de plates-formes de développement en bioinformatique à l'interface des laboratoires de biologie et d'informatique (cf http ://genouest.org). Le chercheur informaticien impliqué en biologie doit travailler pour avoir force de proposition par rapport à la modélisation des phénomènes, tout en apprenant à respecter et prendre en compte la complexité des processus de vérification des hypothèses par les biologistes. Quelques extraits significatifs de ce rapport en rendront mieux compte

[3]Les débats qu'a lancés J. Monod en biologie autour des notions de téléologie et téléonomie sont encore riches mais nous préférons largement cette notion plus neutre de sémantique qui suppose juste que la stabilité est acquise sur des états pourvus d'avantages sélectifs, sans supposer qu'il y ait poursuite d'un but.

que de longs discours :

- "This report identifies four distinct but interrelated roles of computing for biology. 1. Computational tools... 2. Computational models... 3. A computational perspective on biology 4. Cyberinfrastructure and data acquisition."
- "Biological entities are sufficiently complex that it may well be impossible for any human being to keep all of the essential elements in his or her head at once ; if so, it is likely that computers will be the vessel in which biological theories are held, formed, and evaluated... Twenty-first century biology will be an information science... Computing itself can provide biologists with an alternative, and possibly more appropriate, language and sets of intellectual abstractions for creating models and data representations of higher-order interactions, describing biological phenomena, and conceptualizing some characteristics of biological systems."
- "The committee believes that over time, computing will assume an increasing role in the working lives of nearly all biologists... Differences of intellectual style occur because the individuals involved are first and foremost intellectuals. For example, for the computer scientist, the notions of modeling systems and using abstractions are central to his or her work... But many -perhaps most- biologists today have a deep skepticism about theory and models, at least as represented by mathematics-based theory and computational models... The committee... also believes that students of the new biology would benefit greatly from some study of engineering."
- "Advances in biological understanding may yet have enormous value for changing computing paradigms ...but these advances are themselves contingent on work done over a considerably longer time scale... the impact of biology on computing falls much more into the 'high-risk, highpayoff' category."
- "Problem-focused research carries the major advantage that problems offered by nature do not respect disciplinary boundaries... The problem domains discussed in this report include high-fidelity cellular modeling and simulation, the development of a synthetic cell, neural information processing and neural prosthetics, evolutionary biology, computational ecology, models that facilitate individualized medicine, a digital human on which a surgeon can operate virtually, computational theories of self-assembly and self-modification, and a theory of biological information and complexity."
- "The committee believes that 21st century biology will be based on a synergistic mix of reductionist and systems biologies."

J'ai choisi pour ma part de présenter brièvement trois questions-clés, trois axes scientifiques d'ampleur par rapport à ce constat que je ne prétends ni imposer ni résoudre mais qui éclaireront le parcours scientifique présenté dans le reste du document. J'ai travaillé sur les problèmes de modélisation du vivant avec l'hypothèse fondamentale qu'il s'agit de machines symboliques et la volonté d'apporter de l'aide au chercheur en biologie pour traiter avec le bon niveau d'abstraction ces machines symboliques. **Le coeur de mes travaux considère les ensembles de séquences que forment les macromolécules du vivant comme des langages formels et cherche à approfondir les concepts nécessaires pour mener à bien l'analyse linguistique de ces séquences.**

Au delà de cette représentation d'un calcul de nature combinatoire, il y a le constat que les séquences ne constituent pas la totalité du vivant mais simplement sa partie "informationnelle" et que la cellule est en constante interaction avec son environnement. Intégrer l'ensemble des connaissances disponibles, à la fois le niveau informationnel et le niveau biochimique, est une impérieuse nécessité pour avancer dans une compréhension partielle de ce qu'est le vivant. Là encore, je pars de l'hypothèse que l'approche symbolique a tout son sens dans ce contexte où c'est le comportement d'une machine symbolique que l'on cherche à expliquer. Ceci nécessite de mêler à la fois calcul et raisonnement. Les questions que pose la biologie, science expérimentale par excellence, s'expriment majoritairement en termes de raisonnement hypothétique. Si l'on découvre une certaine structure linguistique au niveau des séquences, il lui correspond vraisemblablement un mécanisme, une "algorithmique" biologique associée, qui se traduit par des changements d'état dans certaines conditions pour la cellule. Je propose à la fin de ce document un projet pour commencer à aborder ces questions de manière automatique.

1.1 Challenge 1 : Abstraire les séquences biologiques

L'analyse de séquences, après avoir été l'un des thèmes majeurs de la bioinformatique, tend maintenant à être délaissée pour des études variées sur la biologie des systèmes, où l'on cherche à comprendre globalement le comportement d'un ensemble de composants biologiques. On comprend aisément l'attrait pour la biologie des systèmes, car c'est elle qui permettra de donner un sens au vivant en étudiant les propriétés émergentes des organismes en tant que systèmes. Les biologistes y trouvent leur compte car après une longue phase de réductionnisme, il était important que la physiologie redevienne centrale. Les mathématiciens, automaticiens et informaticiens y retrouvent également un paysage connu, de par leur habitude à étudier des systèmes artificiels.

Cependant, on aurait tort de penser qu'il n'y a plus de problèmes difficiles et intéressants sur le plan de l'analyse des séquences. Celles-ci restent au coeur des propriétés informatives et donc spécifiques du vivant. Elles ont évolué vers une organisation complexe que nous sommes encore loin de pouvoir décrire et expliquer. De même qu'il ne viendrait à l'idée de personne d'étudier un système informatique au niveau du code machine qui le décrit ultimement, on ne peut se contenter de la forme brute des séquences génomiques pour étudier les multiples machineries qu'elles codent.
Les questions principales qui se posent sont alors : **quelles sont les unités de sens au niveau des séquences et comment s'organisent-elles pour former un langage de plus haut niveau ?**

Au-delà de la compréhension de cette organisation vue comme une donnée statique, il faut se poser la question de son **lien avec la dynamique du système vivant**. Les progrès des travaux sur le développement commencent à lever le voile sur les mécanismes possibles d'interaction entre le niveau informationnel discret des séquences et le niveau physique continu qu'impliquent des phénomènes comme la diffusion. Il reste cependant encore de nombreuses hypothèses à tester avant d'élucider le lien étonnant de colinéarité durant les phases précoces du développement des animaux entre l'ordonnancement des gènes sur le chromosome et leur expression dans l'espace et

dans le temps le long de l'axe antéro-postérieur [129].
De plus, la séquence elle-même est susceptible d'évoluer, non seulement au travers de mutations mais également via des mécanismes de transposition qui peuvent être de grande ampleur sur les génomes (e.g. 45% chez l'homme). Modéliser les éléments génétiques mobiles mais aussi leur évolution dans les génomes est un superbe défi.

Enfin, il faut citer le continent largement inexploré que constitue la **métagénomique**, c'est-à-dire l'étude de populations microbiennes à partir de leur matériel génétique global. On sait que la distinction entre organisme pluricellulaire et communauté d'individus d'une espèce ou de plusieurs espèces s'établit au sein d'un continuum[4]. Dans ce domaine, la biologie systémique a une forte composante écologique, avec des composants qui sont des organismes à part entière, et avec une problématique commune d'étude de mécanismes de communication et de coopération entre composants.
Bien sûr, modéliser de tels systèmes communicants est un problème de choix et une source d'inspiration pour l'informaticien confronté à cette même problématique avec les systèmes artificiels qu'il construit. A plus court terme cependant, c'est encore un formidable problème d'analyse de séquences qui prévaut : comment identifier en masse les souches, comment retracer et organiser les unités génomiques échangées entre les organismes, comment connecter la distribution spatiale des souches et celle de leurs unités génétiques ? Autant de questions qu'il faudra résoudre sur des quantités de données dépassant de plusieurs ordres de grandeur celle de l'analyse des génomes.

Notre contribution à ce challenge est de réfléchir à la fois aux unités lexicales qui forment les briques de base des génomes, en particulier en se basant sur l'étude des éléments génétiques mobiles, et aux langages qui permettront d'exprimer pleinement la richesse des structures observables dans les génomes.

1.2 Challenge 2 : Découvrir et modéliser les relations existant entre entités biologiques

La métagénomique est un cas particulier d'étude de l'interaction, une des propriétés fondamentales des êtres vivants sur laquelle se concentrent justement les efforts.
La mode est à la science des réseaux [18], indépendamment de l'origine du réseau (web, réseau métabolique, réseau neuronal, réseau économique...). Ces questions théoriques sont importantes bien sûr, à condition de ne pas escamoter le fait que la question qui préoccupe le biologiste est plus spécifique : savoir quelles particularités révèle l'étude du vivant, qu'il s'agisse de **réseaux de signalisation et de régulation** pour transmettre et contrôler les communications ou de **réseaux métaboliques** pour aboutir à un produit.
Encore bien peu d'études adressent ce point difficile. Peut-on construire une bibliothèque générale de composants élémentaires pour ces réseaux, leur associer des fonctions génériques logiques (in-

[4]Les biofilms constituent un champ d'investigation exceptionnel pour explorer cette émergence de phénomènes mutualisés stables propres au vivant. D'un point de vue fondamental, ceci affine la notion de compétition darwinienne stricte par des possibilités d'adaptation coopérative.

terrupteur, ou exclusif -bistabilité-, oscillateur...), physico-chimiques (régulateur thermique, capteur lumineux, stockage d'oxygène), morphologiques, etc... et raisonner sur leur assemblage ? Quels modèles pour expliquer non pas la dynamique finale mais la dynamique de connection lors de l'évolution de ces réseaux qui a permis d'aboutir au réseau observé [174, 60] ? Peut-on retracer dans les réseaux formés les processus d'adaptation à des éléments environnementaux majeurs [185] ?

Même au niveau moléculaire, on est encore loin de maîtriser l'ensemble de ces phénomènes et il s'agit d'un obstacle majeur pour une compréhension générique des fonctions biologiques élémentaires.
On commence à bien connaître les interactions entre acides nucléiques, que ce soit pour des appariements classiques Watson-Crick ou des constructions plus complexes structurant les ARN. On découvre cependant chaque jour un peu plus la complexité du **monde des ARN** [150], en particuliers des petits ARN, qui avaient largement échappé aux investigations et jouent des rôles variés au sein des cellules.
Le **monde des protéines**, bâti sur des éléments, les acides aminés, beaucoup plus variés que les acides nucléiques, résiste quant à lui très largement à nos tentatives de le comprendre : avec l'hypothèse raisonnable que la structure des protéines dépend largement des séquences qui les constituent, quelle est la logique d'établissement des ponts cystéine, des ponts salins et autres liaisons à l'intérieur des protéines ou entre différentes protéines ? Comment prédire avec le minimum d'information épigénétique les interactions entre protéines ou ARN et ADN, qui interviennent dans la régulation des gènes ? Comment distinguer les domaines ou unités fonctionnelles à l'intérieur des protéines et peut-on déduire quelque chose de la combinatoire de leurs arrangements ?

La réponse à toutes ces questions reposera sur une double avancée, technologique et scientifique, et la modélisation y prendra une large part. Notre contribution à ce challenge est passée par l'utilisation des langages pour la modélisation et l'apprentissage automatique pour la construction des modèles.

1.3 Challenge 3 : Assister l'expérimentation scientifique

La science est actuellement largement dominée par une technologie extrêmement dynamique et inventive en biologie. L'exemple récent de la technologie des puces à ADN, qui a envahi les laboratoires sans aucune préparation méthodologique et scientifique et a généré un gâchis d'études redondantes, incite à réfléchir sur une meilleure coordination entre ces deux niveaux. La technologie a toujours tiré la science en avant mais la biologie pose un problème spécifique : les données expérimentales ont longtemps et sont encore largement considérées comme ayant une valeur intrinsèque, supérieure à celle des modèles. Ceci a la double conséquence néfaste de limiter leur diffusion et de produire en masse des données peu exploitables. C'est pourquoi nous voulons terminer en plaidant pour un interventionnisme bienveillant du mathématicien et de l'informaticien jusqu'au niveau expérimental.

Pour mettre au point un modèle, les biologistes ont essentiellement deux stratégies possibles :

la stratégie pas à pas, qui consiste à faire varier un paramètre à la fois et observer les variations correspondantes du modèle ; et la stratégie empirique, qui consiste à récolter un grand ensemble de données et bâtir une théorie cohérente au-dessus.

Ces stratégies d'expérimentation ne passent simplement pas à l'échelle, face à la complexité des modèles visés.

Les informaticiens l'ont constaté très tôt, on ne peut pas comprendre ou tester le comportement d'un programme uniquement en observant son exécution pas à pas sur un ensemble de données quelconques. Il faut donner au biologiste la possibilité de raisonner de manière plus abstraite sur les connaissances disponibles et l'ensemble des observations, puis l'assister dans la génération des jeux de tests pertinents pour discriminer le plus efficacement possible parmi l'ensemble des hypothèses celles qui sont en accord avec le système. Autrement dit, il faut préparer un nouveau cadre expérimental qui soit à la hauteur des nouveaux défis de la biologie et ceci ne se résoudra pas en multipliant les dispositifs classiques.

Les besoins sociétaux en protection des individus, en protection de leur environnement et en énergie créent d'ores et déjà une forte demande de progression dans le domaine de l'expérimentation, qui va de toute évidence exploser dans les vingt ans à venir (médecine personnalisée, agronomie durable, maîtrise des risques de pollution). Signe parmi d'autres, il y a eu une directive européenne approuvée par le parlement en décembre 2005, REACH[5], qui impose à toute compagnie produisant plus de 10 tonnes de produits chimiques par an d'évaluer leur risque potentiel, directement ou indirectement, pour la santé humaine et pour l'environnement. On estime actuellement qu'il y a un peu plus de 30 000 produits chimiques à tester, avec le seul recours actuel de l'expérimentation animale intensive ! Il faut préparer un cadre d'expérimentation intensive, mieux adapté à la compréhension du vivant. D'une certaine façon, il faut que le biologiste privilégie de plus en plus l'action par rapport à l'observation sur le vivant.

La biologie synthétique, qui cherche à développer des technologies inspirées des systèmes électroniques et informatiques, rendant l'ingénierie biologique plus facile et plus fiable, est un des axes possibles des développements nécessaires. On sait déjà construire des circuits simples : interrupteurs [86], oscillateurs [15] et même maintenant des mémoires [8] ! Du point de vue de la modélisation, ceci contribue non seulement à la compréhension des systèmes biologiques, mais offre également l'accès à des conditions expérimentales sophistiquées sur le vivant, qui peut être alors contrôlé de l'intérieur.

Notre contribution à ce challenge est un projet que nous décrivons en dernière partie de ce document et qui s'intéresse à la conception d'un micro-laboratoire "intelligent". Il s'agit d'étudier les obstacles qui restent à lever pour obtenir un laboratoire biologique tenant sur une lame de microscope, relié à un moteur logiciel d'expérimentations capable de générer de façon partiellement autonome des expérimentations sur ce laboratoire. Un tel projet représente pour nous l'ultime phase d'abstraction nécessaire pour assister pleinement le biologiste dans ses recherches sur les systèmes biologiques.

Les obstacles ne manqueront pas. Certains biologistes auxquels j'ai fait part d'un tel projet sont sceptiques sur le fait qu'on puisse jamais recréer des conditions expérimentales satisfaisantes, proches du in vivo. D'autres craignent que ce type d'étude tue une certaine créativité expérimen-

[5]Registration, Evaluation & Authorisation of Chemicals

tale qui fait le sel de la biologie. Les premiers obstacles ne seront donc pas seulement de nature technologique ou scientifique...
D'autres sciences avant la biologie ont subi les mêmes affres de l'automatisation d'une partie des tâches intellectuelles qui leur incombaient, la physique, la chimie, et plus récemment les mathématiques. Il n'en est jamais résulté un appauvrissement mais au contraire la possibilité de traiter des problèmes plus ambitieux. Si nous réussissons à avancer, il y aura sans doute plus d'ordinateurs et moins de paillasses. Mais ce que j'aime à considérer comme le génie expérimental (je dis volontiers sans connotation péjorative "génie du bricolage"...) du biologiste aura toujours sa place et ses effets seront simplement amplifiés par l'automatisation des tâches les plus routinières.

Chapitre 2

Structures lexicales sur les génomes

Abstraire les séquences, c'est d'abord repérer les unités de sens à l'intérieur de celles-ci, unités qui trahissent leur présence d'abord parce qu'il en existe plusieurs exemplaires répétés le long des séquences. Nous avons étudié à différents niveaux de granularité comment repérer ces unités copiées dans les génomes.
On rejoint ici une problématique proche de la compression, où il s'agit de représenter une information de manière compacte en tirant parti d'une table de mots avec des fréquences élevées. Certains travaux ont d'ailleurs été au bout de cette logique en proposant des schémas de compression de séquences génomiques. Hélas, les séquences génomiques s'avèrent difficiles à comprimer.

Une approche de la recherche de répétitions par des méthodes universelles manquera généralement de finesse dans ses résultats et permettra surtout de retrouver des phénomènes bien connus, repérables manuellement. La raison principale de cet échec relatif est le fait qu'une séquence biologique est tout sauf une séquence aléatoire et que les divers mécanismes de copie qui y sont à l'oeuvre ont superposé leur action tout au long de l'évolution. Pas d'accès aux originaux ici, il faut se contenter de la lecture des génomes actuels, avec toute leur complexité ! Pour progresser dans cette recherche, il faut faire appel à d'autres critères que le simple critère fréquenciel et tirer parti des connaissances ou émettre des hypothèses sur la manière dont ont été générées ces répétitions.

L'accès à l'étude de génomes complets a rapidement montré que leur séquence contient un nombre élevé de répétitions. Celles-ci jouent des rôles majeurs dans la structure, la fonction, la dynamique et l'évolution des génomes, que ce soit dans les archées [27, 153], les bactéries [5, 178] ou les eucaryotes [38, 4, 84, 125] [1]. Notre propre génome est constitué lui-même à plus de 50 % de séquences répétées [International Human Genome Sequencing Consortium, 2001] ! Des études convergentes ont montré que la variation du nombre d'occurrences de certaines répétitions peut représenter un facteur important dans l'apparition de certaines pathologies comme le diabète, l'épilepsie, le retard mental lié au X-fragile, certaines dystrophies musculaires...

[1] La présence de ces répétitions constitue d'ailleurs une difficulté technique majeure pour obtenir la séquence génomique d'un organisme, car cette dernière est constituée à partir de l'assemblage de fragments dont il sera difficile de distinguer l'original et la copie.

L'analyse des éléments génomiques répétés est donc un problème important en bioinformatique.

2.1 Explorer l'ensemble des répétitions d'un génome

La mort n'est pas drôle ... parce qu'elle ne supporte pas la répétition.
Boris Vian

La littérature sur la recherche de répétitions dans les séquences est vaste. Elle s'intéresse tout d'abord naturellement au modèle le plus simple de répétitions, où celles-ci se définissent comme des *facteurs (mots) ayant plusieurs occurrences identiques* dans la séquence S étudiée.
Un certain nombre de travaux confondent mots périodiques et répétitions, exigeant que les occurrences soient contiguës. Une répétition est alors dans sa forme la plus simple le carré d'un facteur et dans sa forme étendue une puissance éventuellement rationnelle d'un facteur (la racine), c'est à dire un mot de la forme $u^n v$, où v est un préfixe de la racine u. Ceci correspond à l'abstraction d'une notion bien particulière de répétition pour les biologistes, appelée répétition en tandem, qu'on trouve effectivement dans les génomes.
Cependant, d'une manière générale, les occurrences peuvent être réparties de manière quelconque dans la séquence, y compris de manière chevauchante.
A la base de la construction de ces répétitions, il y a donc le dictionnaire de l'ensemble des facteurs distincts de S. La taille de ce dictionnaire est une des mesures possibles de la complexité de S. On sait que cette complexité est au pire quadratique en fonction de la taille de la séquence [109]. Il suffit pour l'atteindre de considérer le mot $a^n b^n$ pour un entier n fixé. Pour ce mot, il existe en effet un nombre quadratique de facteurs uniques (les $a^i b^j$, $1 \leq i \leq n, 1 \leq j \leq n$) et un nombre linéaire de facteurs avec au moins deux occurrences (les a^i et les b^i, $1 \leq i \leq n-1$). Le nombre de répétitions au sens où nous venons de le définir est linéaire dans cet exemple. Il suffit cependant de considérer la copie $(a^n b^n)^2$ pour retrouver la complexité maximale quadratique pour les répétitions. On sait également que la complexité demeure quadratique en moyenne lorsque l'on utilise des modèles de mélange de type chaîne de Markov pour la génération des séquences [117].

Pour des raisons de complexité (on doit traiter des séquences de l'ordre de 10^9 caractères, et un comportement quadratique est difficilement acceptable), on a cherché à isoler certaines répétitions à la fois représentatives des autres et en nombre restreint. Le concept le plus remarquable en ce sens est celui de **répétition maximale**, facteur de S qu'on ne peut étendre ni à droite ni à gauche sans en diminuer le nombre d'occurrences dans S. De ce fait, ce concept relie l'entité abstraite du mot (son contenu) à son extension dans la séquence (sa distribution).

Plus généralement, il existe une vaste littérature couvrant le problème de la recherche de répétitions, qui peut être principalement divisé en trois catégories en fonction du type de répétition visée : répétition exacte, répétition approchée ou répétition structurée.

Dans la première catégorie des **répétitions exactes**, on trouve les concepts de répétition directe, en miroir ou palindromique, de répétition la plus longue (longest repeat, [123, 137]), de

répétition maximale (maximal repeat, [99, 130, 131, 181]) et de répétition super maximale (super-maximal repeat [99, 3]).

Définissons plus formellement la notion de *répétition maximale* d'une chaîne S, qui mérite qu'on s'attarde un peu dessus :
Il s'agit d'un mot w tel qu'il existe deux mots awc et bwd de $\$S\$$ avec $a \neq b$ et $c \neq d$ ($ est une lettre n'apparaissant pas dans S).
Les répétitions maximales se distinguent par un bon nombre de propriétés sympathiques :

- Elles sont bien structurées par la relation d'inclusion. Munissons Σ^*, l'ensemble des mots possibles, de la relation d'inclusion sur les mots et munissons N, l'ensemble des positions de S, de la relation d'inclusion sur les ensembles. L'ensemble des couples (w,P_w) de $\Sigma^* \times 2^N$ reliant les répétitions w avec les positions P_w de fin des occurrences de w, forme un treillis de concept au sens de Wille [85]. Cette constatation découle directement de [182].
- Elles sont en nombre linéaire (au plus n répétitions maximales dans une séquence de taille n) tout en représentant l'ensemble des facteurs répétés ;
- Enfin elles peuvent être calculées en temps linéaire, très facilement à partir de la plupart des structures d'index [99, 182, 1, 179].

Cependant, les répétitions maximales souffrent de deux défauts importants vis-à-vis des répétitions «réelles». Tout d'abord, il est difficile en pratique de les distinguer du bruit de fond. Mettre en évidence des répétitions de grande taille conduit généralement à des unités pertinentes car la probabilité qu'elles apparaissent par chance est très faible. Les petits mots au contraire, tels ceux qui apparaissent dans la régulation des gènes, ont une fréquence comparable à la fréquence de mots aléatoirement choisis de taille similaire.
Ensuite, aucun niveau de bruit ou de variation n'est autorisé entre deux copies. En pratique, la plupart des copies partagent une séquence très similaire mais ne sont pas identiques. Nous avons approfondi les implications de cette observation élémentaire sur la distribution des occurrences des répétitions maximales. On peut déjà noter que les répétitions maximales sont définies simplement par rapport à un contexte minimal de leurs occurrences et qu'il est intéressant d'étudier des contextes moins limités.

Dans la deuxième catégorie des répétitions approchées, nous rangeons les algorithmes dont le souci principal est d'introduire un **modèle d'erreur dans la spécification des unités répétées**. On trouve ainsi les plus longues répétitions avec un bloc de jokers (don't care) [145], les paires maximales avec un espacement borné (bounded gap) [32, 132], les répétitions consécutives plus ou moins dégénérées [189, 210, 28] et les répétitions à une distance d'édition près [135].
Ces algorithmes sont ainsi mieux adaptés à l'analyse des nombreuses répétitions génomiques qui contiennent en général des copies avec de multiples variations. Ils peuvent être ainsi utilisés pour localiser des duplications ou n-plication ($n > 1$) de gènes et différents types de répétitions courtes en tandem qui sont, parmi d'autres, des constituants des centromères[2] et des télomères[3].

Enfin le troisième type d'approche considère la **recherche de motifs structurés**, où les répétitions sont constituées d'une collection ordonnée de $p > 1$ parties séparées les unes des autres par les espacements contraints [149, 110, 154, 175]. Ces algorithmes se révèlent particulièrement

[2]Partie centrale d'attachement des chromatides durant la mitose ou la méïose.
[3]Extrémités des chromosomes, intervenant dans leur stabilité.

intéressants dans l'étude de l'expression des gènes et de leur régulation, où des motifs souvent très courts sont impliqués dans l'attachement des macromolécules entre elles. Ils se positionnent à la frontière entre analyse lexicale et analyse syntaxique puisqu'on commence déjà ici à assembler de manière controlée des unités élémentaires.
Le concept clé nous semble celui de *flexibilité*, par opposition à la notion d'*approximation* que nous avons décrit précédemment.
Les approches relatives aux répétitions approchées mettent, l'accent sur l'extension d'une expression (un mot ou un langage) par un modèle général d'erreurs qui est censé prendre en compte toutes les variations possibles autour de cette expression. Dans la recherche de motifs flexibles, l'accent est mis plutôt sur la fragmentation de l'expression en petites sous-expressions qui doivent apparaître de manière exacte, et en leur assemblage par fixation de contraintes sur les distances possibles entre les positions des sous-expressions à l'intérieur des occurrences.
D'un point de vue algorithmique, la recherche de motifs approchés a été très tôt abordée en bioinformatique en se basant sur un filtrage préalable de mots exactement présents, les *graines*, ce qui a assuré un succès durable à l'algorithme de comparaison de séquences Blast [10]. L'idée fondatrice est que pour un nombre limité d'erreurs (p. ex. 2 erreurs dans un mot de taille 15), on trouve forcément des facteurs non chevauchants du mot d'origine de longueur raisonnable (p. ex. 1 mot de taille 5 ou 3 mots de taille 3 dans le cas précédent). Ce filtrage de motifs exacts a été étendu plus récemment avec des notions de graines espacées et de graines multiples [211, 169, 77, 107], qui correspondent à cette recherche de flexibilité.
D'un point de vue biologique, dès lors que l'on considère la séquence comme porteuse d'information et pas simplement réduite à une fonction chimique, il est tout à fait vraisemblable que les mutations aléatoires produites soient d'une certaine façon contrôlée au cours de l'évolution et que la présence de ces facteurs communs soit exploitée.

Si l'on cherche à synthétiser ces apports fertiles, *une répétition*, telle qu'elle apparaît en biologie, se réfère alors à une entité composée d'une *liste ordonnée de mots d'un langage fixé, apparaissant à des distances contraintes*. Ces mots sont eux-mêmes des éléments apparaissant en plusieurs copies dans la séquence de manière parfaitement conservée.

Tous les modèles développés jusqu'à présent restent encore limités en expressivité par rapport aux formes de répétition connues dans les séquences biologiques. On peut retrouver celles-ci dans différentes banques publiques, regroupées en familles présentes dans un grand nombre de génomes ou au contraire dans une espèce particulière [120, 121, 188, 26].
Dans la littérature en biologie, on peut trouver la description de trois grandes classes de répétitions : *les répétitions en tandem* (copies consécutives de motifs), *les segments dupliqués* (qui incluent les duplications de gènes et de segments chromosomiques) et les *éléments mobiles* (sur lesquels nous avons travaillé, incluant le cas particulier important des transposons). PILER [73], qui représente probablement l'état de l'art courant dans la classification des types de répétitions biologiques, distingue quatre groupes : TA, DF, PS et TR. Les classes TA (tandem array) et DF (dispersed family) correspondent aux répétitions en tandem et dispersées. Les deux autres classes sont les pseudosatellites (PS), éléments regroupés dans les génomes qui ne sont pas des répétitions en tandem et les répétitions terminales (TR), qui sont des copies d'un même élément localisées à la fin d'un élément dupliqué.

– Les répétitions en tandem proviennent vraisemblablement d'un dérapage d'un chromosome

répliqué glissant sur son motif. Les motifs dans les répétitions en tandem sont des k-mères, k valant généralement moins de 5 (micro-satellites), mais pouvant atteindre des valeurs beaucoup plus grandes (jusqu'à plusieurs milliers, avec une taille totale pouvant couvrir plusieurs mégabases). Le nombre exact de répétitions dépend des individus, et est d'ailleurs utilisé comme empreinte digitale dans les recherches de paternité. Les microsatellites (STR) ont un motif de taille 2 à 10 et une taille globale de moins de 200 paires de bases. Les minisatellites ont un motif de taille 10 à 100 et une taille globale de moins de 1kb à 20kb. Ils sont généralement riches en dinucléotides GC.

– Les segments dupliqués sont de larges portions d'ADN inter ou intra-chromosomiques d'une taille globale de 40 à 700 kb, qui résultent vraisemblablement d'accidents de réplication dupliquant des groupes entiers de gènes.
– Enfin, les éléments mobiles (aussi appelés "répétitions dispersées" ou "interspersed repeats") sont des séquences d'ADN qui, comme leur nom l'indique, sont parsemées le long du génome et sont produites par différents mécanismes, souvent d'auto-réplication, qui sont un élément majeur d'évolution des génomes.
  Ils ont été découverts par B. McClintock dans les années 1940 chez le maïs et on ne cesse depuis lors d'en trouver dans toutes sortes d'organismes. Chez les mammifères, les éléments mobiles les plus communs sont les LINEs en inter-chromosomique (longueur de l'ordre de 6-7 kb) et les SINEs en intra-chromosomique (longueur de l'ordre de 300 bases).

Cependant on est probablement encore loin d'avoir découvert tous les types de répétitions présentes dans les génomes.

Le reste du chapitre évoque nos travaux autour des répétitions dans le but de distinguer les unités de sens du génome.

Toute répétition observée dans une macromolécule, est le résultat d'un processus de duplication qui peut être dû à différents mécanismes, et ceci mène à différentes classes de répétitions qui n'ont très probablement pas encore été toutes découvertes. Nous avons commencé par étudier la manière de **visualiser ces répétitions** pour explorer librement leur typologie. La difficulté était l'explosion potentielle des relations à considérer entre les paires d'occurrences représentant une même copie. Nos travaux nous ont conduits à passer d'une structure d'exploration arborescente à une structure pyramidale, qui nous semble plus adaptée à la structure mathématique de ces répétitions maximales.

Excepté pour le domaine des répétitions en tandem où il y a maintenant une compréhension en profondeur des modèles nécessaires, des travaux supplémentaires sont nécessaires afin de mieux comprendre la nature des répétitions biologiques.
Nous avons étudié les extensions possibles du cadre de travail formel habituel sur les répétitions dans une séquence. La difficulté est d'**améliorer le réalisme de la modélisation (les caractéristiques majeures) des répétitions observées dans le contexte de la biologie moléculaire, tout en conservant la généralité des concepts**.

L'enjeu est de développer des critères sélectifs qui ne dépendent pas de la taille des répétitions, un seuil difficile à fixer en pratique. Ainsi, nous avons déjà évoqué le problème de la détection de mots courts répétés, difficiles à dégager de la masse de leurs occurrences dans les génomes. Ce qui

rend ces petites copies pertinentes est le fait qu'elles apparaissent généralement dans le voisinage d'une position initiale particulière. Les éléments répétés sont souvent regroupés en structure compacte et nous avons introduit une notion formelle de *localité* pour décrire une telle contrainte. Une notion complémentaire, nécessaire à prendre en compte, est la présence d'éléments indirectement locaux parce qu'ils sont dans le voisinage d'éléments locaux. Nous avons proposé la notion de *répétition voisine* pour exprimer une telle contrainte.
D'une manière générale, étudier la manière de contraindre les distances entre occurrences de répétitions ouvre le chemin d'une nouvelle plaine de jeux pour l'algorithmique sur les mots dont les travaux sur les motifs structurés ont fourni les premières investigations.

Nous avons également travaillé sur le concept fondateur de *répétition maximale* en abordant trois problèmes. Le premier est celui de l'exploration pratique de la masse des occurrences détectées dans un génome. Le second problème est celui de l'étude de l'impact de petites variations sur les répétitions maximales. En particulier nous nous sommes intéressés au cas des SNP (Single Nucleotide Polymorphism), phénomène très fréquent dans les séquences biologiques où une base est localement mutée dans une copie d'un élément dupliqué.

Enfin, nous nous sommes attaqués aux variations les plus générales des copies, qu'il est intéressant de considérer en termes d'assemblage de blocs élémentaires. Nous avons ainsi réfléchi sur les *notions d'unités (sur une séquence) et de modules (sur un ensemble de séquences)*, définis comme des composants de base sur les macromolécules d'ADN.

Il semble probable que les concepts développés ne soient pas spécifiques au domaine des séquences biologiques et puissent être trouvés dans de nombreuses applications où les séquences ne sont pas générées de manière aléatoire et où les occurrences de répétitions apparaissent à des positions corrélées (langage naturel, compression de données...). La délimitation des répétitions peut alors être effectuée en fonction de leur environnement et en fonction de leur distribution dans les séquences.

2.2 Visualisation de répétitions

Lorsqu'on cherche à visualiser l'ensemble des mots d'une séquence, le premier choix à fixer est celui de la forme du dictionnaire, pas simplement pour une exploration automatique mais également pour une exploration manuelle. Conceptuellement, la structure de données d'arbre compact des suffixes permet de décrire facilement la structuration de l'ensemble des facteurs d'une séquence, et de mettre au point rapidement des algorithmes de recherche en se basant sur cette structure d'arbre. Elle n'est pas parfaite mais reste pour moi la structure la plus équilibrée dans le compromis efficacité/intelligibilité. Le tableau de suffixes par exemple, qui reprend simplement la notion classique de dictionnaire comme liste de mots triés par ordre lexicographique, perd la structure naturelle d'arbre qui permet de sauter facilement d'un préfixe à l'autre. Les différentes variantes d'automates, où l'on cherche à compacter certains facteurs communs, restent l'alternative la plus intéressante mais perdent l'avantage de permettre de se diriger progressivement vers une position donnée de la séquence.

Nous rappelons qu'un arbre compact des suffixes pour une séquence S est un arbre dont les

arcs sont étiquetés par des mots non vides, dont tous les noeuds internes ont au moins deux fils, et tel que chaque suffixe de S corresponde à exactement un chemin depuis la racine de l'arbre jusqu'à une feuille. Chaque noeud peut être associé avec le mot composé à l'aide des lettres lues sur le chemin depuis la racine jusqu'à ce noeud. La construction d'un tel arbre sur une séquence S peut être effectuée en temps et espace linéaire en fonction de la taille de S (voir [91] pour une revue de qualité).

Nous avons travaillé il y a une douzaine d'années de cela sur cette structure de données pour montrer son intérêt non seulement du point de vue algorithmique mais également d'un point de vue modélisation, pour permettre aux biologistes d'explorer de manière plus abstraite la structure des génomes. Ceci a donné lieu à la thèse de R. Gras [97] et au développement d'un premier outil, Forest, de visualisation et d'analyse des grandes séquences biologiques [96].
À l'époque de cette publication, la version disponible de la banque nucléique d'EMBL (Release 44) comprenait 500 000 séquences et un peu plus de 360 millions de bases et justifiait déjà la mise en place de nouveaux outils d'abstraction du contenu linguistique disponible. La version de fin 2007 atteint 111 millions de séquences et 196 milliards de bases. La progression exponentielle des séquences est toujours de mise et la taille moyenne de ces séquences est également en augmentation.
S'il était encore possible au début du séquençage de remarquer des structures étonnantes au niveau des séquences, cela n'est maintenant plus réalisable sans assistance . Plus que jamais, il reste crucial de pouvoir naviguer dans ce corpus de textes du vivant en ayant une vision rapide et hiérarchisée de l'information qu'ils contiennent, d'un point de vue simplement lexical.

Bien sûr, les outils ne manquent pas pour parcourir un génome et ses annotations. Dans le cadre des grands programmes de génomique, ils correspondent à des développements relativement lourds. Créés essentiellement pour étudier les génomes des vertébrés, les plus utilisés actuellement sont le navigateur de génomes Ensembl [106] et le navigateur de génomes du NCBI [122, 134]. Leur but est d'intégrer les données de génomique à la fois de manière verticale (sources et types de données multiples) et horizontale (organismes multiples).
La base de données sous-jacente (MySQL) permet d'effectuer de nombreuses requêtes, mais la navigation dans la séquence reste avant tout "positionnelle", avec plusieurs niveaux de zoom sur les chromosomes. Il semble très difficile de découvrir de nouvelles caractéristiques avec ce type de visualisation dont l'usage principal est d'observer la distribution ou la corrélation locale d'annotations préalablement renseignées.
On dispose par ailleurs maintenant d'outils de visualisation de génomes spécialisés, capables de produire des images de qualité "publication" des données attachées à des séquences génomiques de prokaryote ou d'eukaryote : ChromoWheel [75], CGviewer [209] ou Circos [133], qui permettent une vision plus synthétique des données et donc leur exploration pour l'établissement d'hypothèses scientifiques. Cependant, tous ces outils n'exploitent pas la séquence elle-même mais uniquement la représentation linéaire ou circulaire de l'ensemble de ses positions.

Nous avons proposé d'utiliser l'arbre des suffixes comme structure de visualisation, donnant accès à la fois à un dictionnaire hiérarchique de toutes les répétitions exactes présentes dans la séquence, et aux positions de la séquence, qui sont en bijection avec les feuilles de l'arbre. L'arbre comme structure de données permet des calculs d'attributs synthétisés associés aux mots

de l'arbre, qui peuvent être effectués en temps linéaire sur l'ensemble des mots. Bien sûr, l'arbre peut être accessible uniquement au niveau logique, les calculs effectifs pouvant être menés sur des structures plus compactes comme le tableau de suffixes [2].

Le calcul des attributs s'effectue au moyen d'une procédure générique fondamentale, que nous notons **Synthèse**. Celle-ci permet de décrire à un niveau plus abstrait la plupart des algorithmes sur cette structure, tout en garantissant une construction de complexité linéaire en temps et espace par rapport à la taille de la séquence analysée. Bien que nous n'introduisions aucun schéma algorithmique nouveau dans cette procédure, c'est la première fois à notre connaissance qu'elle est explicitée pour permettre la formalisation et l'exploitation systématique de la structure d'arbre des suffixes, une structure ubiquitaire "aux mille vertus" [13].

La procédure **Synthèse**, présentée dans l'Algorithme 1, modifie récursivement un attribut sur l'ensemble des noeuds rencontrés durant un parcours en profondeur d'abord de l'arbre. Pendant ce parcours, les données sont synthétisées à partir des fils. Ces données sont stockées dans une variable appelée $Resultat$ (ligne 6). Une fois que tous les fils ont été récursivement traités, l'attribut du noeud $Node$ est calculé par la procédure $Attr$ en fonction des données stockées dans $Resultat$ (line 8). La fonction $estFeuille$ permet de tester si un noeud est une feuille. Suivant qu'un noeud $Node$ est terminal ou pas, on utilise deux fonctions d'initialisation de données différentes : initFeuille pour les feuilles (ligne 2) et initNode pour les autres noeuds (ligne 4).

Algorithm 1 Calcul générique d'attributs sur un arbre des suffixes

Synthèse($Node$, initNode(), update(), initFeuille(), Attr())

```
1:  if estFeuille(Node) then
2:    initFeuille(Node, Resultat)
3:  else
4:    initNode(Resultat) ;
5:    for each Fils of Node do
6:      update( Resultat,
                Synthèse(Fils, initNode, update, initFeuille, Attr ) )
7:    end for ;
8:    Attr(Node, Resultat)
9:  end if ;
10: return Resultat
```

Pour un exemple simple d'application de cette procédure, on peut considérer le calcul classique des répétitions maximales, dont on discute plus avant dans la section 2.4. Celui-ci peut être effectué grâce à l'appel suivant :

$$Synthèse(root, empty, addLetter, previousLetter, isDiscriminant),$$

où

- $Resultat$ représente l'ensemble des contextes gauches, c'est-à-dire, l'ensemble des lettres avant la position de chaque occurrence du mot associé au noeud courant.
- $empty$ initialise $Resultat$ avec l'ensemble vide ;
- $addLetter$ est décrit dans l'Algorithme 2 ;

– $previousLetter$ met $S[p-1]$ dans $Resultat$, où p est la position de la feuille dans S.
– $is\_discriminant$ indique si $Resultat$ contient au moins deux lettres différentes.

Algorithm 2 Gère l'ensemble des lettres précédant le mot courant dans l'arbre des suffixes

$addLetter$(Resultat, ResultatFils)

1: **if** (Resultat=ResultatFils) **or** (Resultat=$\emptyset$) **then**
2: Resultat $\leftarrow$ ResultatFils
3: **else**
4: Resultat $\leftarrow \top$ %au moins deux lettres différentes comme contextes gauches
5: **end if**

Le logiciel Forest est fondé sur ce calcul générique d'attributs et sur une utilisation conjointe de dictionnaires et de cartes.
On autorise le parcours de l'espace des mots par affichage d'un sous-arbre de profondeur fixée (à partir de la racine ou d'un noeud sur lequel on a cliqué), en se focalisant sur ceux qui possèdent des caractéristiques remarquables calculées à partir d'attributs.
On peut ensuite établir des cartes de la distribution de mots ou de motifs sélectionnés dans la séquence. Chaque carte est associée à un vecteur sur les positions de la séquence, et on peut combiner ces vecteurs à l'aide d'opérateurs booléens.
Le but était alors pour nous de proposer un prototype démontrant la possibilité de calculs intensifs à la demande dans une interface d'exploration agréable. Nous avions illustré les possibilités de Forest sur l'analyse du génome des bactéries *Bacillus subtilis* et *Escherichia coli* [97].
La figure 2.1 donne un aperçu des écrans interactifs produits par le logiciel. On montre une partie

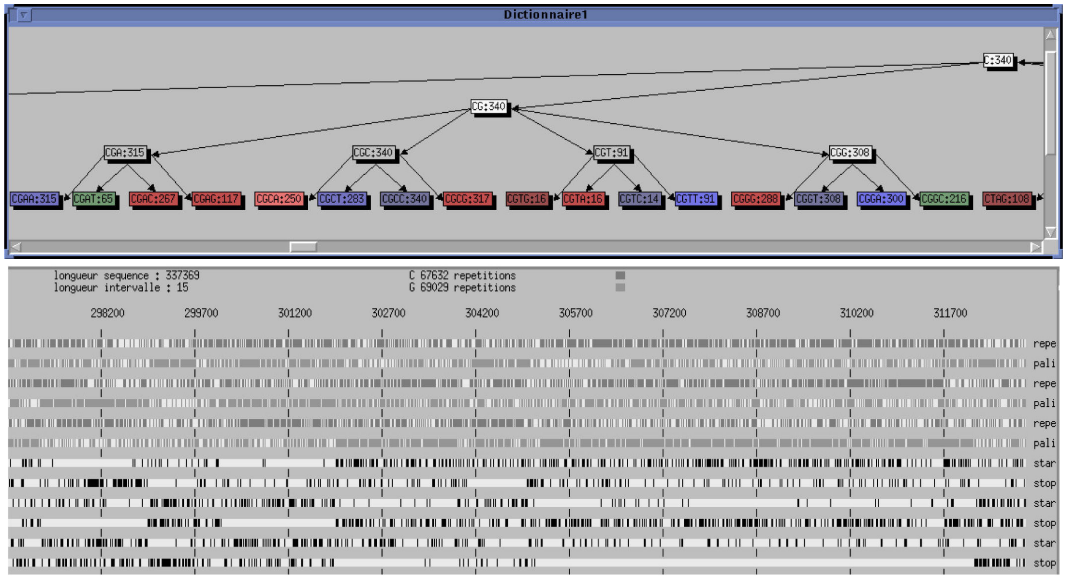


FIG. 2.1 – Représentations associées à l'analyse des répétitions d'un contig de 337k de *B. subtilis.*

de l'arbre produit sur *B. subtilis* sur lequel deux attributs ont été mis en valeur. Le premier donne la taille de la plus longue répétition sous un noeud (nombre associé au noeud) et le second indique la fréquence du mot dans le génome à l'aide d'un code de couleurs. On montre également une carte de répartition d'un mot et de son palindrome dans les six phases de lecture de la séquence, en même temps que la présence de codons start et stop.

L'usage principal que nous envisagions pour ce type d'outils est d'offrir un support expérimental pour des études théoriques sur l'usage du code dans les génomes. Concevoir un outil pleinement fonctionnel, en particulier pour l'étendre à un traitement multi-génomes et y intégrer les sources communes d'annotations, aurait nécessité un travail trop important en regard de cet usage limité.
Nous avons donc mené une deuxième série d'investigations pour obtenir un outil d'utilité élargie. Le point faible de Forest réside dans la difficulté d'observer simultanément un nombre important de répétitions, qui peuvent partager éventuellement des facteurs communs et se chevaucher sur les cartes. En particulier, il est très difficile avec Forest de visualiser des variations autour d'une répétition donnée. Il est clair également que l'affluence de génomes et de familles d'éléments partageant le même rôle au sein d'un ou plusieurs génomes rend les **visualisations comparatives entre différentes séquences** particulièrement importantes. Nous avons donc travaillé autour de ces deux concepts : relation entre répétitions maximales et intégration des données multi-génomiques.

Afin de mieux cerner les difficultés, commençons par une revue rapide des principaux outils de visualisation disponibles pour analyser ces énormes quantités de répétitions génomiques.
Parmi les plus simples et les plus utilisés, il faut d'abord citer les "dotplots" [90], qui ont de multiples variantes et sont de simples affichages dans le plan de la matrice de comparaison entre deux séquences (éventuellement identiques). Un point est affiché à la coordonnée (X, Y) si le mot de taille fixée situé à la position X dans la séquence 1 est similaire (avec un seuil fixé) au mot à la position Y dans la séquence 2. Ce domaine a été très actif, important des techniques utilisées par ailleurs dans la visualisation de données scientifiques et bien d'autres représentations existent : les paysages [41, 54], les jeux du chaos [118], les "PIP" (Percent Identity Plots) [194], les graphes de répétitions [135] et les "BARD" (Biological ARc Diagrams) [207] pour ne citer que les principaux. L'interprétation des vues créées par ces méthodes est assez délicate, en particulier pour les grands génomes, du fait que la plupart reposent sur l'affichage de paires d'occurrences de répétitions. Des représentations comme les dotplots, les jeux du chaos ou les BARD sont conçus essentiellement pour des comparaisons de positions de répétitions dans des paires de séquences génomiques. Elles deviennent difficiles à utiliser quand la taille des séquences ou le nombre de répétitions augmentent, en particulier parce qu'elles ne permettent pas de percevoir les structures d'inclusion entre différentes répétitions. Enfin, les outils proposés restent souvent au stade de prototype et manque d'options pratiques importantes pour leur utilisation, comme l'agrandissement de régions d'intérêt.

Nous avons proposé avec P. Durand un nouvel outil de visualisation spécialisé dans l'étude des occurrences de répétitions, qui associe les fonctions de visualisation et d'interrogation sur les répétitions maximales d'une ou de plusieurs séquences. Il est basé sur un concept appelé "diagramme pyramide" ou simplement Pygram [69], qui propose d'afficher toutes les répétitions maximales exactes localisées à l'intérieur d'une même séquence ou entre plusieurs séquences, sans se préoc-

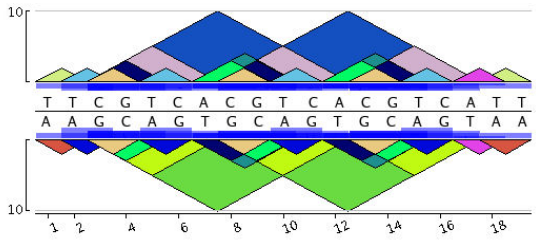
cuper des liens existant entre paires de répétitions. Ces Pygram peuvent être considérés comme une reconstruction rationnelle de la notion de paysage [41].

Un Pygram repose sur une intéressante propriété de l'ensemble des répétitions maximales vis-à-vis de l'inclusion sur les mots :

proposition 1. *Supposons que deux répétitions maximales A et B se superposent, c'est-à-dire qu'il existe un mot non vide W tel que W est un suffixe de A et un préfixe de B, alors cette intersection W est une répétition maximale.*

On peut donc non seulement visualiser les répétitions mais également leur organisation hiérarchique, qui d'après la proposition précédente n'est pas un arbre mais une pyramide [23]. Il suffit de prendre soin d'afficher les plus petits mots après les plus grands[4]. En contraste avec les autres outils que nous avons évoqués, Pygram ne cherche pas à afficher des paires de répétitions. En conséquence, il peut produire une vue pertinente des séquences répétées à tous les niveaux, depuis la séquence génomique entière jusqu'au niveau des nucléotides [68].

Si l'on veut préciser un tout petit peu les choses, un Pygram pour une séquence génomique *S* de taille *n* est un graphique à deux dimensions dont nous donnons un exemple ci-contre, où *S* et toutes ses répétitions maximales (MR) sont indiquées le long de l'axe des *x*.



Plus précisément encore, étant donnés des facteurs d'agrandissement k_x et k_y pour les 2 axes, le $i^{ème}$ nucléotide de *S* est à la position (*i/k*,0), et la MR de taille *m* à la position *i* dans *S* correspond à l'intervalle [*i/k*,(*i*+*m*)/*k*] sur l'axe des *x*. La taille *m* de la MR à la position *i* dans *S* est symbolisée par un triangle isocèle (**une pyramide**) de hauteur $\delta m/l$. δ est soit '+1' pour une répétition positionnée sur le brin normal (N), ou '-1' pour une répétition positionnée sur le brin complémentaire inverse (RC) de *S*.

On trouvera un exemple de Pygram sur un génome d'archée en Figure 2.8. Pygram semble particulièrement utile dans l'exploration des génomes de virus, de plasmides, d'archées et de bactéries, qui sont extrêmement nombreux[5] et sur lesquels on doit pouvoir mener une exploration rapide.

Nous avons mis au point récemment un nouvel outil pour tenir compte de l'importance de la visualisation multi-génomes (publication en cours). L'affichage simultané de plusieurs génomes impose des fonctionnalités nouvelles de navigation et de sélection. L'idée principale est d'introduire une possibilité de synchronisation au niveau du défilement des séquences. On peut ainsi

- Sélectionner un ensemble de mots soit dans le dictionnaire, soit dans une séquence de référence, et afficher la première occurrence disponible sur chacune des séquences ;
- Se déplacer sur l'ensemble des séquences de manière synchronisée ou non, en sautant éventuellement d'occurrence en occurrence ;

[4] La représentation pyramidale a par ailleurs été utilisée pour la classification sur les séquences génomiques [177].

[5] Au 1er mars 2008, on comptait 52 archées, 600 bactéries, 580 plasmides et 1480 virus dont le génome était complètement séquencé.

– Sélectionner une suite de mots dans une séquence de référence et rechercher cette suite, en respectant un espacement similaire entre les mots, sur l'ensemble des séquences.

La figure 2.2 donne une illustration des capacités de ce nouveau logiciel. On y aperçoit, en bas, un panneau de contrôle permettant de sélectionner les répétitions, la suite de répétitions, et/ou les séquences à afficher ; au milieu, une séquence de référence où apparaît en rouge la sélection d'une suite de répétitions ; en haut, l'ensemble des séquences (il s'agit de différentes souches d'une bactérie), avec un calage synchronisé sur la première occurrence d'une suite de répétitions similaire à celle sélectionnée. Les mots sont des copies exactes, par contre, les distances peuvent varier légèrement. Enfin, le panneau de gauche donne différentes informations sur chaque séquence, dont le nombre de copies d'une telle structure (il y en a par exemple 1 dans la première et 4 dans la troisième).

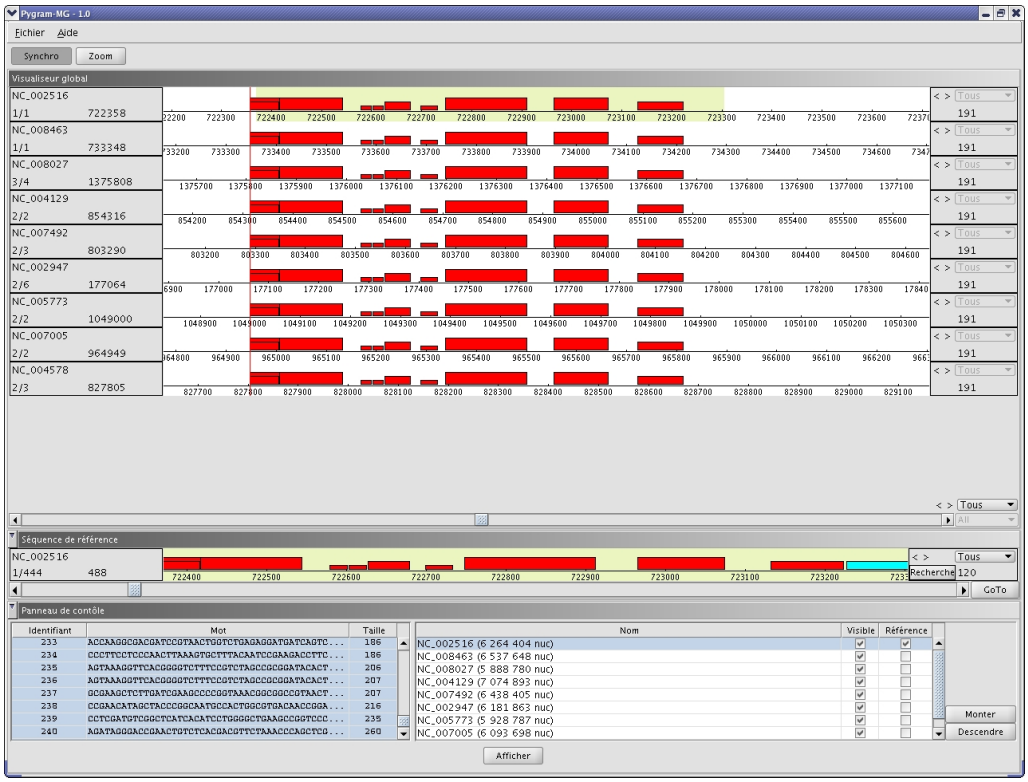


FIG. 2.2 – Affichage synchronisé de plusieurs génomes de souches d'une même espèce, *Pseudomonas*

Nous pensons actuellement que les outils de modélisation nécessaires au biologiste exigent des capacités d'analyse structurelle encore plus élevées et qu'un outil lexical générique doit surtout être conçu comme un moteur de recherche efficace piloté par un analyseur syntaxique. C'est la voie dans laquelle nous nous sommes engagés.

Cependant, ces études lexicales restent indispensables pour préciser quelles sont les unités élémentaires intéressantes à considérer dans les génomes. Les répétitions maximales sont un point

de départ solide pour cela, mais il faut aller au-delà pour sélectionner un nombre raisonnable *d'occurrences* et non pas seulement *de mots*. Rappelons en effet que la borne sur le nombre linéaire de répétitions maximales ne dit rien sur le nombre d'occurrences de ces répétitions. Ceci est important en pratique lorsqu'on souhaite regarder leur distribution dans une séquence. En fait, leur nombre peut être quadratique en fonction de la taille de la séquence. Par exemple, la séquence CA^nGA^nT admet exactement n répétitions maximales, tous les A^k, pour $k = 1 \dots n$. Le nombre de leurs occurrences est $\sum_{i=1}^{n} 2i = n(n+1)$.

Nous avons recherché des critères qui permettent de filtrer efficacement ces occurrences sans recourir systématiquement au critère de taille choisi dans la plupart des études. Un des plus intéressants est celui de localité.

2.3 Localité des répétitions

A la suite de plusieurs études, et notamment d'un travail sur la formalisation des structures de CRISPR dans les séquences prokaryotiques, nous avons développé avec C. Rousseau, A. Siegel, P. Peterlongo et F. Coste (article en cours de soumission) la notion de *Localité*, qui vise à contraindre de manière simple la distribution des répétitions dans une séquence. Les CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) sont des unités répétées présentes dans la plupart des espèces d'archées et dans de nombreuses bactéries [153]. Du fait de leur variabilité, elles sont utilisées pour le génotypage en épidémiologie. Elles jouent vraisemblablement un rôle important dans l'immunité microbienne et semblent agir à travers un mécanisme de type interférence d'ARN. Malgré de nombreuses études, aucune définition précise des CRISPR n'est actuellement disponible, et ceci est clairement un frein pour établir pleinement les hypothèses scientifiques sur leur structure et leur rôle dans les génomes microbiens.

Pour contraindre la distribution d'un mot dans une séquence, l'idée fondamentale est de limiter la distance maximale entre deux de ses copies. On a en fait besoin d'une notion un peu plus raffinée qui permette de regrouper les occurrences en classes et de regarder les distances à l'intérieur de ces classes.

Définition de la portée et de la localité

Nous commençons par introduire la *portée* d'un mot, qui donne accès à la taille des régions d'occurrence du mot dans les séquences. De façon typique, cette portée aura une valeur faible pour des répétitions regroupées.

définition 1. (**Portée**) La *portée* d'un ensemble d'occurrences dans une séquence est la différence entre la plus grande et la plus petite position de cet ensemble.

La portée est une statistique simple reliée au caractère aléatoire d'une distribution : une distribution étroite est typique de mots particulièrement intéressants. L'inverse n'est pas vrai. Pour des séquences biologiques, on ne peut se contenter de définir la portée d'une répétition comme celle de ses occurrences car il peut exister plusieurs groupes éloignés bien distincts pour un même mot dans un génome. Pour de telles distributions multimodales, la portée pourra être artificiellement large. C'est pourquoi nous introduisons un paramètre μ, le nombre maximal de groupes

autorisé par mot, chacun groupe comportant au moins deux copies. La portée de l'ensemble sera la moyenne de la portée de chacun des groupes.

définition 2. **(μ-portée)** Étant donné un entier μ, une *μ-partition* est une partition qui contient au plus μ parties, chacune avec au moins deux éléments. Soit I l'ensemble des positions des occurrences d'un mot W dans une séquence S. On considère une μ-partition de I qui minimise la somme des portées de l'ensemble des parties puis on définit la *μ-portée* de W comme la valeur moyenne des portées dans cette partition optimale. Formellement,

$$\mu\_portée(W) \quad = \quad \frac{\min_{\mu\_partition(P)} \left\{ \sum_{i=1}^{|P|} (\max P_i - \min P_i) \right\}}{|P|},$$

La portée globale d'un mot est donc calculée pour une partition optimale du point de vue de l'ensemble des positions de tous les clusters. Un mot non répété est un mot dont la μ-portée est nulle.

Nous pouvons maintenant introduire le critère de localité.

définition 3. **(μ-localité de répétition)** Étant donnée une séquence S, une répétition a une *μ-localité* λ dans S si son ensemble d'occurrences a une μ-portée au plus λ dans S. Étant donné un ensemble de séquences $\mathcal{S}$, une répétition a une *μ-localité* λ dans $\mathcal{S}$ si elle a une μ-localité λ sur chaque séquence de $\mathcal{S}$.

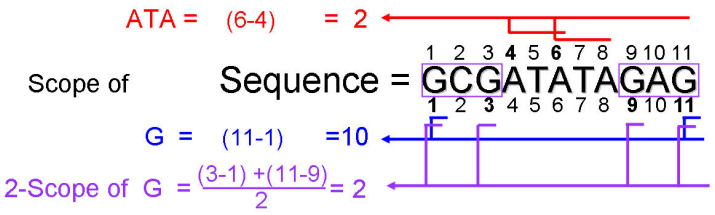


FIG. 2.3 – Portée d'un facteur répété

La figure 2.3 illustre le calcul de portée sur la séquence $S = GCGATATAGAG$. La portée de G est 10, la portée de A est 4, la portée de ATA, T ou TA est 2. La 2-portée de G est $(2+2)/2 = 2$, correspondant à la partition $\{\{1, 3\}, \{9, 11\}\}$ de son ensemble d'occurrences.

Soit $S = ATATATCCCCCCCATGAT$. La lettre C est un facteur de 1-localité 5 et de 2-localité 2. Le facteur ATA a une 1-localité λ pour $\lambda \geq 2$. AT a une 1-localité λ pour tout $\lambda \geq 15$ et une μ-localité λ pour tout $\mu \geq 2$ et $\lambda \geq 3.5$.

Notons que la définition de portée n'est pas réduite à des mots et pourrait être étendue à des motifs puisque seules les positions sont considérées.
L'accroissement de μ n'augmente jamais la valeur de la portée et celle-ci converge vers l'intervalle moyen entre deux occurrences consécutives, reflet de la façon dont elles sont groupées. Nous définissons *la localité asymptotique* ou simplement *la localité d'une répétition* comme sa μ-localité quand μ tend vers l'infini.

Énumérer toutes les partitions possibles des occurrences d'un mot n'est pas une approche envisageable pour le calcul de la portée. Nous avons démontré deux propriétés qui permettent en pratique un calcul linéaire de celle-ci en fonction du nombre d'occurrences.

proposition 2. *Soit $\mu \in \mathbb{N}$ et W une répétition avec n occurrences dans une séquence $pos[1..n]$ ($n \geq 2$). La* μ-portée *de W est égale à*

$$\mu scope(W) = \frac{pos[n] - pos[1] - opt(\mu, n)}{\nu},$$

où il existe une partition $\{P_1, \cdots P_\nu\}$ de $pos[1..n]$, $\nu \leq \mu$ telle que

- *chaque bloc P_i correspond à un intervalle sur $pos[1..n]$ qui contient au moins deux positions, a_i et b_i : $a_i = \min P_i < b_i = \max P_i < a_{i+1}$;*
- *les extrémités des blocs satisfont la relation $\sum_{i=1}^{\nu-1}(a_{i+1} - b_i) = opt(\mu, n)$;*
- *la fonction opt satisfait la formule de récurrence suivante :*

$$\begin{aligned} &opt(1, j) = opt(k, 2) = opt(k, 3) = 0 \\ &opt(k+1, j+2) = \max \left\{ \begin{array}{l} opt(k+1, j+1), \\ opt(k, j) + pos[j+1] - pos[j] \end{array} \right. \\ &\qquad\qquad (k \geq 1,\ j \geq 2). \end{aligned} \tag{2.1}$$

proposition 3. *Soit $\mu \in \mathbb{N}$ et W une répétition avec n occurrences dans une séquence $pos[1..n]$ ($n \geq 2$). La* portée asymptotique *de W est égale à*

$$limitScope(W) = \frac{pos[n] - pos[1] - opt(n)}{\nu},$$

où il existe une partition $\{P_1, \cdots P_\nu\}$ of $pos[1..n]$, $\nu \leq \mu$ telle que

- *chaque bloc P_i correspond à un intervalle sur $pos[1..n]$ qui contient au moins deux positions, a_i et b_i : $a_i = \min P_i < b_i = \max P_i < a_{i+1}$;*
- *les extrémités des blocs satisfont la relation $\sum_{i=1}^{\nu-1}(a_{i+1} - b_i) = opt(n)$;*
- *la fonction opt satisfait la formule de récurrence suivante :*

$$\begin{aligned} &opt(2) = opt(3) = 0 \\ &opt(4) = pos[3] - pos[2] \\ &opt(j+3) = \max \left\{ \begin{array}{l} opt(j) + pos[j+1] - pos[j] \\ opt(j+1) + pos[j+2] - pos[j+1] \end{array} \right. \\ &\qquad\qquad (j \geq 2). \end{aligned} \tag{2.2}$$

Notons qu'un mot qui n'a pas la μ-localité λ peut très bien contenir dans le cas général des facteurs qui ont la μ-localité λ. Ceci est dû à la nécessité d'avoir deux occurrences par bloc dans la partition des occurrences (il suffit de considérer par exemple dans $ATATATCCCACAT$, AT et A avec $\mu = 2$ et $\lambda = 3$).

Calcul de la portée de tous les facteurs répétés d'une séquence

La **1-portée** des facteurs d'une séquence peut être calculée en temps et espace linéaire en fonction de la taille n de la séquence.
Le calcul de la μ-portée et de la portée limite de tous les facteurs d'une séquence a une complexité de O(n^2) en temps et O(n) en espace. Sur le séquences testées, la μ-portée converge en pratique vers la portée asymptotique pour des valeurs de l'ordre d'une dizaine.

Nous donnons juste ici l'algorithme de calcul de la 1-portée, en utilisant la fonction générique Synthèse que nous avons déjà présentée (Algorithme 1). Les autres algorithmes utilisent de la même manière l'arbre des suffixes et développent à chaque noeud un calcul par programmation dynamique respectant les équations de récurrence décrites.

Le calcul de la 1-portée s'effectue par

$$Synthèse(root, initMinMax, updateMinMax, positionMinMax, oneScope)$$

qui range dans $Resultat$ (voir l'Algorithme 1) les positions minimales et maximales des occurrences d'un facteur en utilisant les fonctions suivantes :

- $initMinMax$ donne à $Resultat.min$ la valeur $+\infty$ et à $Resultat.max$ la valeur 0 ;
- $updateMinMax$ met à jour $Resultat$ sur un noeud, avec les positions min et max des feuilles sous un fils donné de ce noeud.
- $positionMinMax$ met dans $Resultat.min$ et $Resultat.max$ la position de la feuille ;
- $oneScope$ range dans $Node.scope$ la valeur de $Resultat.max - Resultat.min$.

Nous avons étudié différentes bactéries et archées, dont le génome de *S. solfataricus*, une archée qui vit près des fumerolles sulfurées de certains terrains volcaniques, à des températures de l'ordre de 80°C et dans un environnement très acide. La figure 2.4 donne un aperçu de la valeur de la μ_portée sur la séquence génomique de ce microbe.

On montre à la fois les répétitions simples et les répétitions maximales de taille comprise entre 18 et 80, et on fait varier μ de 1 à 10. Les expériences ont également été menées sur une version brouillée du génome ("shuffle" des séquences afin d'obtenir une séquence aléatoire de même composition en lettres) et on a vérifié que les courbes restaient plates dans ce cas. Par contre, dans le cas des génomes réels, les courbes obtenues sont très dépendantes des espèces analysées.

L'effet de μ est particulièrement visible sur des courbes 2D affichant la variation du nombre de répétitions maximales et du nombre de leurs occurrences pour un intervalle fixé de portée. La Figure 2.5, réalisée à partir de l'étude de trois bactéries, illustre à quel point les profils différent suivant les espèces. Les variations dans la dérivée de ces fonctions pointent sur d'intéressantes régions avec des répétitions regroupées.

Calcul du δ-voisinage d'un langage dans une séquence

Il existe une autre façon de poser une contrainte sur les distributions de mots dans une séquence. Certains mots sont particulièrement intéressants de façon indirecte, parce qu'ils sont toujours associés à des mots locaux. Nous avons formalisé cette notion dans le cadre le plus général. Le but est de pouvoir rechercher des mots dont les positions sont corrélées avec les positions

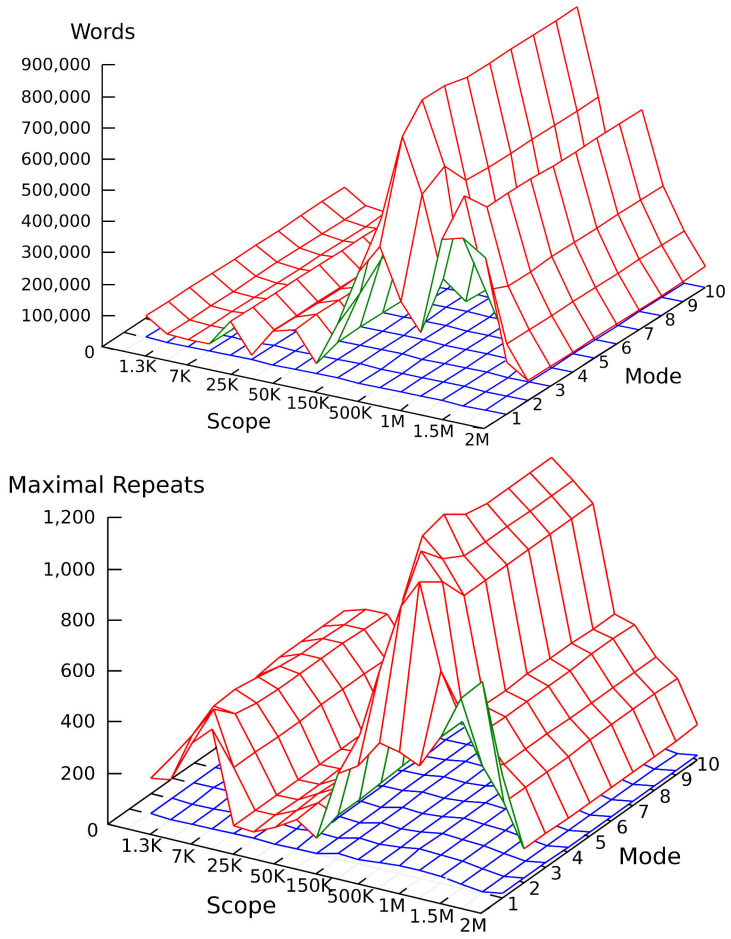


FIG. 2.4 – Étude de l'archée *S. solfataricus. Nombre de répétitions simples et maximales en fonction de la μ_portée*

d'un ensemble donné de mots et le concept vise donc particulièrement les applications où l'on recherche des dépendances entre mots à l'intérieur d'une séquence. Ceci nous semble particulièrement utile dans le cadre de la modélisation syntaxique où nous nous situons, afin de dégager des structures élémentaires sur le lexique observé.

définition 4. (δ-voisin d'un langage dans une séquence) Un δ-voisin (avec support τ) d'un langage L dans une séquence S est un mot tel que toutes ses occurrences dans S occupent des positions qui n'intersectent pas avec les positions de L est tel que toutes (ou au moins τ pourcent de) ses occurrences sont séparées par au plus δ lettres d'une occurrence d'un mot de L. On peut parler de δ-voisinage pour l'ensemble des δ-voisins.

Par exemple, considérons la séquence S et le langage L suivant :

$$S = CTCCCCTTACCTTATTCATTCCTC, \;\; L = \{AT, TA\}.$$

Les mots T ou TT ne sont pas des 1-voisins de L car certaines de leurs occurrences intersectent avec les mots de L. Le mot C n'est pas un 1-voisin de L car il a cinq occurrences à la bonne

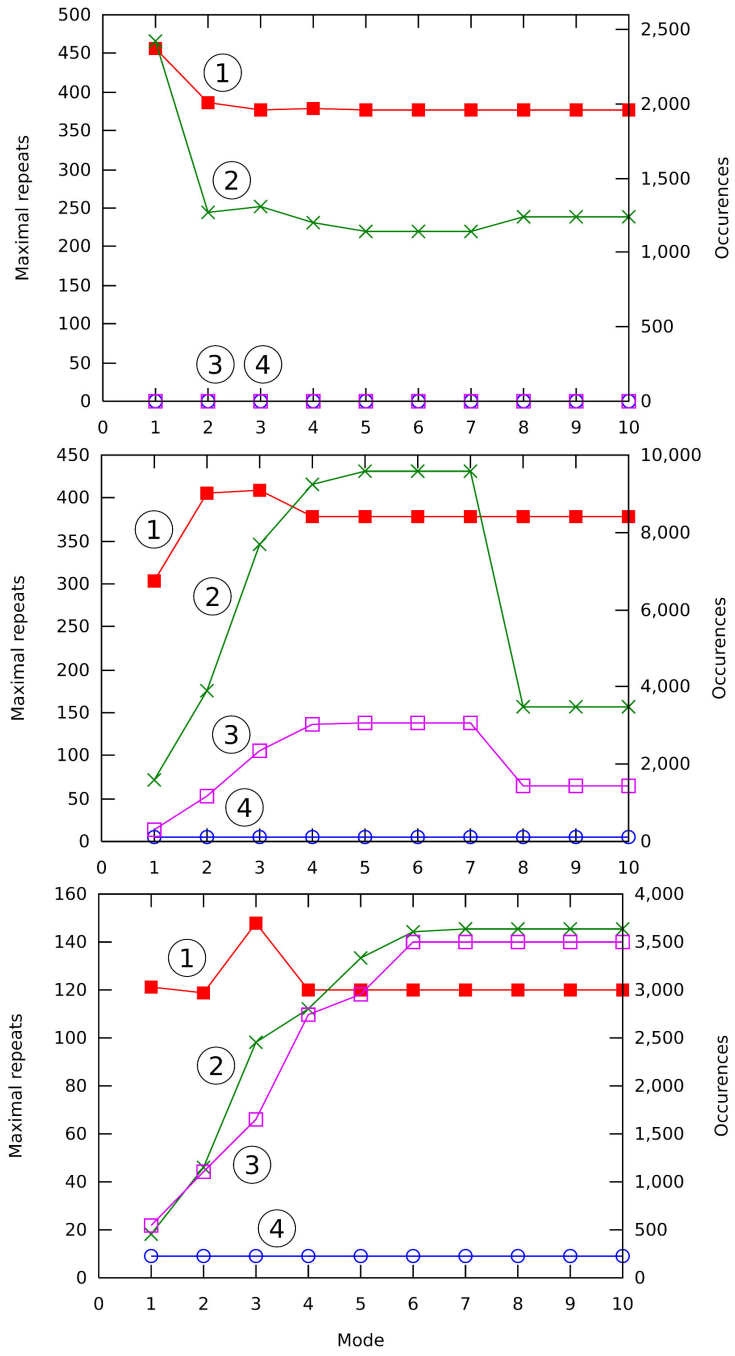


FIG. 2.5 – Étude des bactéries *D. psychrophila* (portée 3 à 7Kb), *G. kaustophilus HTA426* (portée 500 à 750K) et *G. thermodenitrificans NG80-2* (portée 300 à 500Kb). Nombre d'occurrences de répétitions (courbe 2, et courbe 3 pour la version "shuffled") et nombre de MR (courbe 1, et courbe 4 pour la version "shuffled") en fonction du nombre de modes.

distance de L et 6 occurrences qui sont trop éloignées des mots de L. Donc c'est un 1-voisin de L seulement pour un support τ inférieur à 45%. Les mots CC et CCT sont les 1-voisins répétés

avec support 50% de L (CCT est aussi une répétition maximale).

Nous avons introduit la notion de support pour prendre en compte le fait que certaines occurrences des mots référents peuvent avoir disparu et qu'en conséquence des voisins peuvent être observés sans référents associés.

La propriété de voisinage doit vérifier deux conditions : une sur la distance maximale δ à une occurrence de L ; l'autre sur le fait de ne chevaucher aucune occurrence de L. Nous avons proposé un algorithme qui considère dans l'ordre croissant les différentes paires de positions adjacentes dans L et vérifie pour chaque mot apparaissant dans l'intervalle de ces positions que ces conditions sont bien remplies. Si L est de taille p et S de taille n, la complexité de cet algorithme est en o($n^2 + n \log p$) en temps et o(n) en espace.

Nous l'avons testé sur des génomes variés. Nous donnons un aperçu des résultats sur la figure 2.6 pour les répétitions maximales de S. solfataricus, une archée d'une taille de 3Mb qui contient plus de 1.5M de répétitions maximales avec plus de 32M d'occurrences. Les éléments de L ont été fixés comme les 135 répétitions maximales des plus remarquables, d'une taille supérieure à 400. Nous avons ensuite considéré les répétitions maximales voisines. Un des points marquants est que même pour un quorum τ de 100% et une distance faible $\delta = 2Kb$, un seuil très exigeant qui demande que tous les mots considérés aient une distribution incluse dans un voisinage proche de la distribution sélectionnée, de nombreux mots voisins ont été trouvés. Le nombre d'occurrences de voisins est 11235 pour $\tau = 100\%$ et 16851 pour $\tau = 75\%$. Ni les grandes répétitions, ni leurs voisins n'ont une distribution uniforme le long du chromosome. Nous n'avons pas encore mené d'étude significative de ces résultats, mais les voisins peuvent être la trace de motifs de régulation ou de gènes. Les résultats pour un seuil $\tau = 75\%$ soulignent la tendance de "points chauds" dans la distribution, avec un nombre de régions contenant plus de 100 occurrences qui passe de 32 pour $\tau = 100\%$ à 70 pour $\tau = 75\%$. De plus, la taille des mots suit une distribution large allant de mots très petits (taille 8) à de très grands mots (taille maximale 387).

2.4 Répétitions maximales avec des contextes étendus

Les briques de base des répétitions sont des mots délimités par leurs contextes et formalisés en terme de *maximalité*. Le *contexte* d'une répétition dans une séquence correspond aux lettres flanquantes d'une occurrence de cette répétition. La maximalité exprime que les mots sont sélectionnés sur la base d'au moins deux occurrences avec des contextes gauche et droit différents. Les expérimentations sur les séquences génomiques nous ont poussés à étendre cette notion standard de maximalité, en étudiant des contextes qui peuvent ne pas être réduits à une seule lettre, voire appartenir à plusieurs mots.

Contextes et contextes discriminants

définition 5. (contexte d'un mot dans une séquence) Un *contexte* d'un mot W dans une séquence S est une paire de mots (l, r) -les contextes gauche et droit- telle que lWr est un facteur de S.

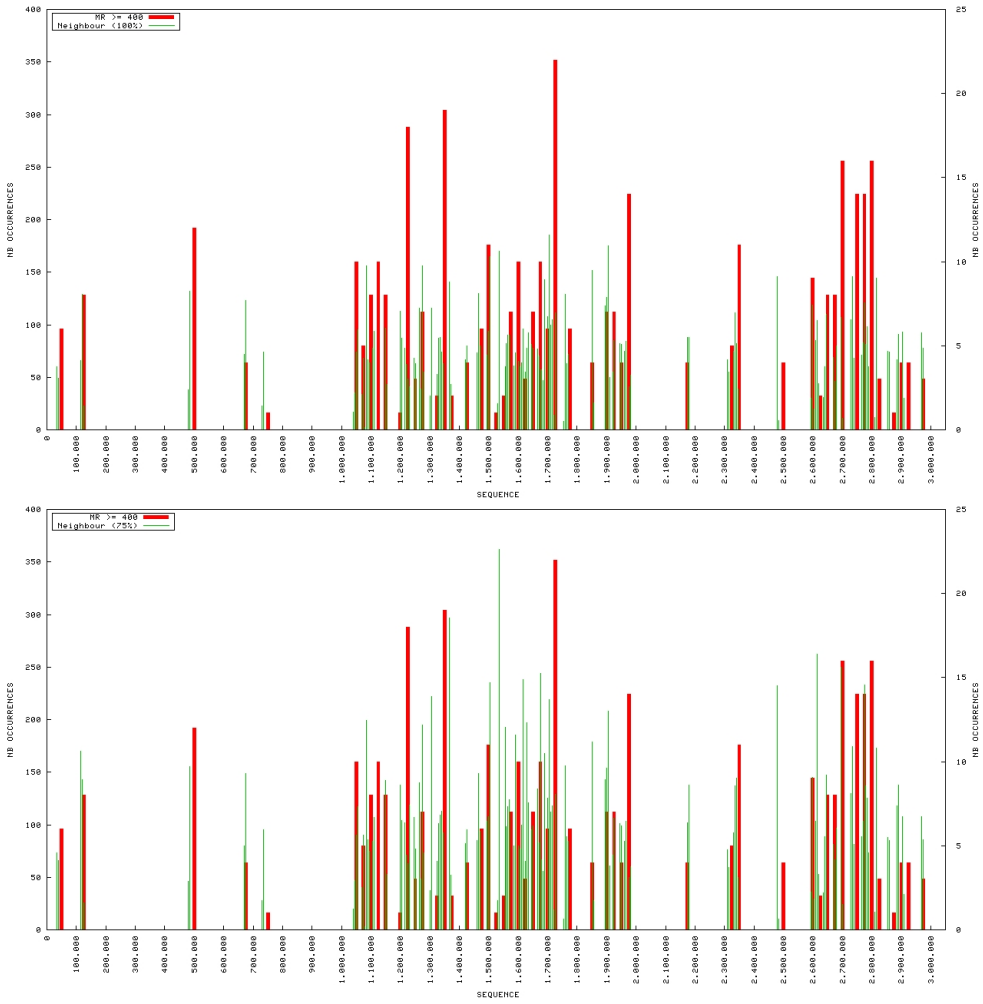
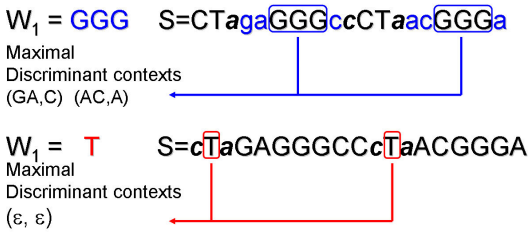


FIG. 2.6 – Étude de l'archée *S. solfataricus*. Nombre d'occurrences de répétitions maximales de taille supérieure à 400bp (en rouge, échelle à droite) et nombre d'occurrences de 2Kb-voisins le long de la séquence (en vert, échelle à gauche) pour un support de 100% et 75%.

Parmi les contextes, ceux qui diffèrent d'une occurrence à l'autre d'un même mot sont particulièrement intéressants.

définition 6. (contexte discriminant) Un contexte$(l[1, s_l], r[1, s_r])$ est un *contexte discriminant* par rapport à un mot W dans une séquence S si et seulement si il existe un contexte de W $(l'[1, s_l], r'[1, s_r])$ tel que : $\forall i \in [1, s_l], \forall j \in [1, s_r]$ $l[i] \neq l'[i]$ et $r[j] \neq r'[j]$. (s_l, s_r) est la *taille* du contexte.



Ces définitions sont illustrées sur le schéma ci-contre. Un mot non répété a un contexte discriminant vide -contexte de taille (0,0)- mais un mot peut avoir plusieurs copies et un contexte discriminant vide comme on peut le voir sur la figure pour le mot T.

Les répétitions maximales sont des mots avec un contexte discriminant de taille (1,1). La taille maximale d'un contexte discriminant pour un mot caractérise d'une certaine manière à quel point il se détache du fond en tant qu'entité pertinente distincte. Ceci permet de définir une extension naturelle de la notion de répétition maximale, plus robuste par rapport aux variations dans les copies des répétitions.

Le problème des micro-variations dans les répétitions

Dès lors qu'on cherche à traiter les approximations dans les copies des répétitions, on tombe plus ou moins directement sur l'intégration de distances d'édition et la recherche autour d'un point d'ancrage par programmation dynamique. Pour des petites variations, on peut espérer aboutir à des algorithmes plus performants. Nous avons commencé ainsi à nous intéresser au cas largement répandu des points de mutations isolés stables (single point mutations, SNP), une source importante de variations individuelles (polymorphisme). Les individus d'une même espèce partagent un génome très proche qui justifie les programmes de séquençage d'espèces. Les variations individuelles qui concernent au moins 10% de la population sont estimées à 0.1 % chez l'homme, ce qui permet néanmoins de trouver un SNP toutes les 300 bases environ dans un génome d'individu [47]. Beaucoup semblent ne pas avoir d'implications fonctionnelles. Celles se trouvant dans les régions codantes et dans les régions régulatrices des gènes permettent par exemple de réaliser des cartographies comparées d'individus présentant ou non certains états pathologiques.
On peut de la même façon s'intéresser aux petites variations intra-génomiques pour des éléments répétés.

Certaines répétitions maximales sont robustes par rapport à ces variations et sont particulièrement intéressantes pour délimiter les frontières des régions répétées :

définition 7. (répétitions k-maximales -kMR-) Si k et k' sont des nombres entiers strictement positifs, une *répétition (k,k')-maximale* dans une séquence est un mot avec un contexte discriminant de taille (k, k').

Prenons par exemple la séquence $TTCAGCATGCT$. Le mot CA est une répétition (2,1)-maximale car (TT, G) et (AG, T) sont des contextes discriminants. Cependant C est une répétition maximale qui n'est pas une répétition (2,1)-maximale car ses contextes sont (TT, A), (AG, A) et (TG, T) et ne peuvent pas être discriminés deux à deux. On peut montrer en fait qu'un tel comportement est caractéristique des SNPs.

proposition 4. *Les anciennes répétitions maximales qui ont été interrompues au cours de l'évolution par un point de mutation isolée correspondent aux occurrences de répétitions maximales qui ne sont pas (2,1)-maximales (ou symétriquement, pas (1,2)-maximales).*

Une conséquence pratique importante de cette proposition est qu'il est possible de reconstruire les répétitions maximales présentant une telle position de mutation interne.

Nous avons recherché les SNP à l'aide du protocole suivant : À chaque position de mutation p correspond au moins une répétition maximale v qui n'est pas (2,1)-maximale, démarrant à la position $p+1$. On choisit le contexte de mots (u, v) la caractérisant tel que

1. La taille de v est maximale ;
2. Le mot u est une répétition maximale et la taille de u est maximale par rapport à v.

On peut ensuite sélectionner les positions de SNP sur la base de la taille de uv. Dans la figure 2.7, on a fixé la taille minimale de uv à 18, une valeur qui conduit à très peu de SNP dans des séquences génomiques mélangées aléatoirement. La taille de uv est affichée le long du chromosome de *Sulfolobus solfataricus* et de *Geobacillus thermodenitrificans*.

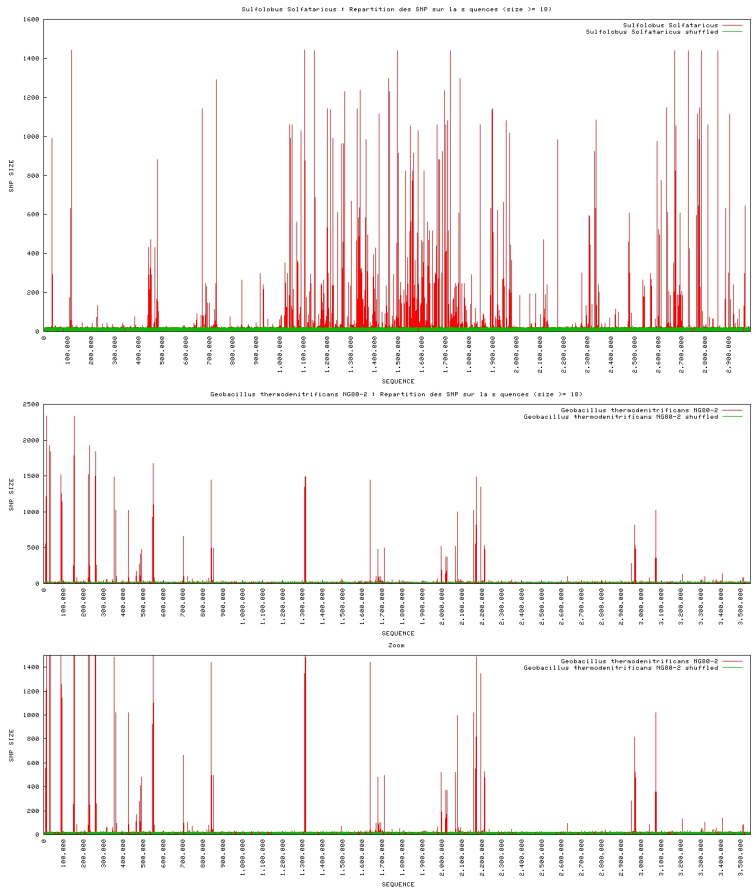


FIG. 2.7 – Taille des contextes répétés($\geq$ 18) encadrant une SNP dans l'archée NC_{002754} (S. solfataricus) et la bactérie NC_{009328} (G. thermodenitrificans NG80-2).

De toute évidence, les distributions observées ne sont pas aléatoires et peuvent différer beaucoup d'une espèce à l'autre. Nous ne faisons ici qu'entrouvrir une porte qui pourrait mener à une

étude plus systématique des variations au niveau intra ou inter-génomes. Une des premières extensions à étudier serait de chercher à maximiser directement la taille de uv à chaque position, un problème qui semble nettement plus difficile qu'une optimisation en deux étapes.

Calculer la liste des (2,1)-répétitions maximales

La détermination des (2,1)-répétitions maximales ne requiert pratiquement pas d'espace supplémentaire par rapport au calcul des simples répétitions maximales avec une structure de données de type arbre des suffixes. Il suffit de gérer les contextes gauches qui apparaissent aux feuilles et de calculer pour chaque noeud l'union des contextes de ses enfants. On mémorise les contextes en les indexant sur leur première lettre et en conservant pour chacune soit la seconde lettre soit le fait qu'il y ait plusieurs secondes lettres possibles. Le calcul peut ensuite être effectué de manière linéaire par l'appel suivant :

$$Synthèse(root, empty, cumulatedcontexte, previousLetters, isDiscriminant),$$

où

- $Resultat$ est composé d'un booléen Resultat.21MR indiquant si le mot est un (2,1)-MR et une fonction représentant l'ensemble des contextes gauches.
- $empty$ initialise $Resultat(\sigma)$ à l'ensemble vide pour toutes les lettres σ ;
- $cumulatedContext$ est décrit dans l'Algorithme 3.
- $previousLetters$ retourne dans $Resultat$ le contexte gauche du suffixe w correspondant à la feuille : si abw est un suffixe, $Resultat(a)$ prend la valeur b et les autres valeurs de la fonction sont mises à $\perp$.
- $isDiscriminant$ retourne la valeur de Resultat.21MR.

Nous laissons ouvert le problème de recherche de (k, k')-répétitions maximales pour des valeurs de k et k' supérieures à 2. Malheureusement, il ne semble pas facile d'obtenir un algorithme itératif sur une taille de contexte croissante. Clairement, la structure de données d'arbre de suffixes atteint ses limites dans ce cas, en particulier pour la recherche de contextes droits. Le problème principal reste celui de la gestion de la taille de l'ensemble des contextes pour une taille $k > 2$.

2.5 Unités et modules

La dernière section de ce chapitre aborde les assemblages élémentaires de répétitions exactes qui sont observables dans les séquences génomiques. Nous avons déjà introduit la notion de flexibilité pour traduire ce point de vue qui privilégie la reconnaissance d'une architecture plutôt que d'un modèle d'erreurs. Deux niveaux d'abstraction sont possibles, un niveau très fin où l'on cherche à assembler des éléments chevauchants, et un niveau plus élevé où l'on cherche à se rapprocher des segments ayant été effectivement copiés dans leur ensemble au cours de l'évolution.

Algorithm 3 met à jour l'ensemble des lettres précédant le mot pointé dans l'arbre des suffixes.

$cumulatedContext$(Resultat, ResultatFils)

```
1: Resultat.21MR← false ;
2: for a ∈ Σ do
3:   if ResultatFils[a]≠ ∅ then
4:     while ¬(Resultat.21MR) and (a' ∈ Σ − {a}) do
5:       if (Resultat[a']≠ ∅) and (Resultat[a']≠ ResultatFils[a]) then
6:         Resultat.21MR← true
7:       end if
8:     end while
9:   end if
10: end for
11: for a ∈ Σ do
12:   if ResultatFils[a]≠ ∅ then
13:     if (Resultat[a]=ResultatFils[a]) or (Resultat[a]=∅) then
14:       Resultat[a] ← ResultatFils[a]
15:     else
16:       Resultat[a] ← ⊤
17:     end if
18:   end if
19: end for
```

Micro-variations et Répétitions chevauchantes

Nous venons de voir que les petites variations sur les répétitions maximales tendent à les fractionner en répétitions maximales de taille inférieure. Essayons de formaliser un peu cette observation. Soit u et v deux mots et a, a', b, b', c et c' des lettres telles que $a \neq a'$, $b \neq b'$ et $c \neq c'$. Considérons un contexte discriminant pour deux occurrences du mot ubv, ($aubvc$, $a'ubvc'$). En cas de substitution de b par b', il est possible d'obtenir à la place la paire ($aubvc$, $a'ub'vc'$) et l'effet est bien de fractionner la répétition maximale en deux mots plus petits u et v. On obtiendra le même comportement en cas d'insertion/suppression.

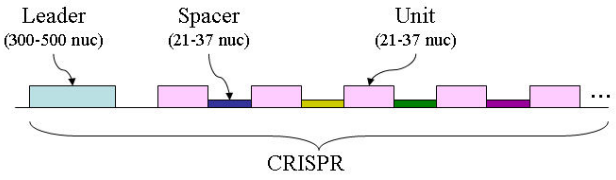
En pratique cependant, on observe dans les séquences génomiques des trains de répétitions maximales chevauchantes plutôt que des répétitions clairement séparées. Ceci peut être assez simplement expliqué par une généralisation du raisonnement précédent, qui introduit simultanément plusieurs points de modification sur des copies différentes. Étant donnée une séquence $uxvyw$, deux points de modification différents génèrent deux séquences $ux'vyw$ et $uxvy'w$, et ceci conduit à deux répétitions maximales chevauchantes uxv et vyw.

Les répétitions en tandem exactes sont elles-mêmes généralement observables comme deux répétitions maximales chevauchantes dans les séquences. En effet, ces répétitions périodiques vont être en contexte de la forme $a(cwb)^k d$ où $a \neq b$, $c \neq d$ et $k > 2$. Dans ce cas, $(cwb)^{k-1}$ sera une répétition maximale qui couvrira de manière chevauchante $(cwb)^k$. Les répétitions en tandem approximatives (ou "avec évolution", [98]) peuvent présenter naturellement des chevauchements. Même sans mutations, toute séquence périodique peut présenter des chevauchements sur chacun

de ses bords. Il est fort possible que des mécanismes de régulation exploitent ces effets de chevauchements, mais à notre connaissance aucune étude systématique n'a été menée dans ce domaine. Nous avons introduit dans le langage de modélisation que nous proposons dans le chapitre suivant cette possibilité d'exprimer des chevauchements.

Nous pensons qu'il est utile dans le cadre de séquences biologiques d'étendre cette notion de répétition maximale chevauchante d'unités de sens, en autorisant le regroupement de mots différents.

Nous illustrons cet intérêt sur une grande famille d'éléments répétés que nous avons étudiée, les CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats), une structure que l'on trouve dans les génomes d'archées et la plupart des génomes de bactéries [116].
Comme on peut le voir sur la figure ci-contre, les CRISPR sont constitués d'un squelette de courtes séquences de 21 à 37 nucléotides de long, répétées presque exactement, que l'on nomme 'unités', séparées par des séquences non répétées de taille similaire appelées 'spacers'.



Dans la plupart des espèces on trouve d'autres éléments associés, comme des séquences 'leader' de 300-500 nucléotides et des gènes (CAS). Dans la Figure 2.8, on montre ce que devient la structure de CRISPR du point de vue de l'affichage des répétitions maximales. Chaque unité de

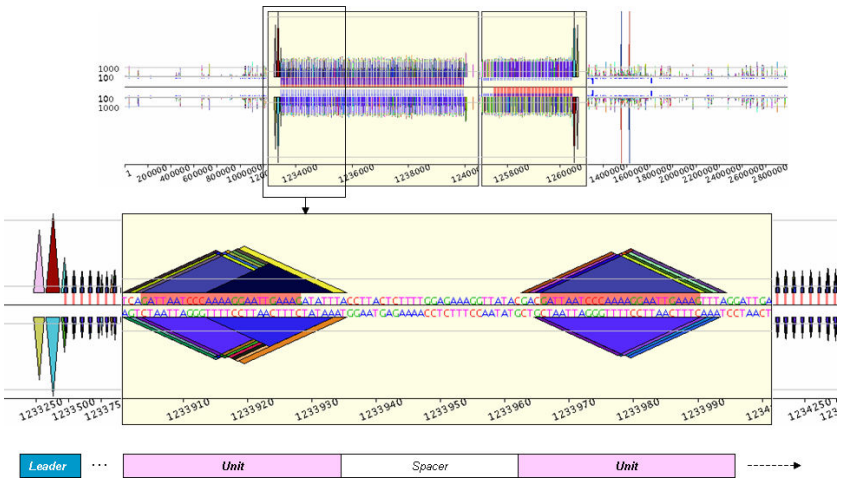


FIG. 2.8 – Un CRISPR vu par Pygram, comme composition de répétitions maximales chevauchantes.

CRISPR y apparaît clairement comme un groupe de répétitions chevauchantes co-occurrentes qui diffèrent simplement par quelques substitutions denucléotides.

Macro-variations et Modules élémentaires

J'ai également étudié avec S. Tempel le cas de variations plus importantes entre deux copies d'une même entité que nous avons baptisée module. Nous avons lancé en 2006 un projet ANR, visant à concevoir des méthodes pour la description de structures de génomes en terme d'assemblage de modules qui sont dupliqués le long d'un génome ou entre plusieurs génomes.
L'architecture de ces modules peut être relativement complexe. Le problème de la décomposition des séquences biologiques en domaines a déjà été bien étudié, au prix de calculs intensifs, pour les séquences de protéines [205]. Cette tâche demeure largement à accomplir dans le cas des génomes et pour ajouter notre contribution à l'inflation des "omiques", nous appelons cette recherche exhaustive de domaines génomiques le "Modulome". Il s'agit d'un problème combinatoire par nature, car considérer les répétitions plutôt que les séquences elles-mêmes conduit à une augmentation rapide du nombre d'objets à analyser : chaque position de la séquence peut contribuer à plusieurs répétitions chevauchantes ou imbriquées. Il y a là un champ fertile pour les méthodes d'optimisation combinatoire, ce que nous illustrons ici sur une méthode de segmentation que nous avons mise au point dans le cadre de la thèse de S. Tempel [216].

En fait, le problème de la segmentation de séquences d'ADN en unités pertinentes a été étudié dès que des séquences ont été disponibles, et continue à être un problème d'actualité, mais ceci s'est effectué chaque fois sur un type bien précis de sous-séquences. On peut citer la détection d'isochores, zones de composition homogène en bases [40, 101] ; d'îlots CpG ("C proceeds G"), zones non méthylées riches en dinucléotide GC en amont de nombreux gènes [146, 100] ; d'origines et de terminaisons de réplication ou encore la séparation des régions codant pour des gènes des régions non codantes [79, 141, 142].
La grande majorité de ces méthodes est basée sur une analyse en composition de sous-séquences de taille fixée, en utilisant des statistiques variées.
Les deux techniques de base sont

- Une estimation de modèles de Markov cachés et une segmentation effectuée en étiquetant les segments avec le meilleur modèle le long de la séquence [57].
- Une segmentation récursive binaire, procédant de façon descendante en partant de la séquence entière et en trouvant récursivement la meilleure séparation en deux sous-séquences en calculant deux indices, un pour le classement des choix de coupure (e.g. divergence de Jensen-Shannon) et l'autre pour décider de l'arrêt de la récursivité (e.g. basé sur la complexité ou sur un test statistique par rapport à des séquences aléatoires) [17, 22, 141, 162].

Toutes ces études ont en commun de rechercher une décomposition brute des séquences en se basant sur leur contenu statistique global ou local. Cependant, il est possible de considérer le problème de la segmentation à un niveau plus fin, où les segments sont caractérisés par l'ensemble des mots (c'est-à-dire les sous-séquences) qu'ils représentent. En d'autres termes, **la segmentation peut être effectuée en fonction d'un langage**.

Dans tous les cas, la question peut être posée dans un cadre formel commun.
Analyser l'architecture en domaines d'une séquence peut être ramené à un problème que Gionis et Mannila ont proposé récemment comme problème générique d'optimisation qu'ils nomment problème de la (k, h)-segmentation [92]. Il consiste à trouver la meilleure segmentation d'une

séquence S de longueur n en k segments, chaque segment provenant d'un sous-ensemble C_h à h éléments d'un ensemble de sources C, avec $h < k$.
On suppose qu'on sait calculer $P(s/c)$ pour tout $s \in S$ et tout $c \in C$ et on cherche à maximiser, sur l'ensemble des sous-ensembles $C_h \subseteq C$ et l'ensemble des affectations $\{c_1^j, ..., c_k^j\}$ dans C_h, le produit $\prod_{i=1}^{k} P(s_i/c_i^j)$. Ce problème a été montré NP-difficile sous des conditions assez larges. En particulier, la programmation dynamique n'est pas applicable car le choix d'une affectation à une position influe sur les choix possibles aux autres positions.

J'ai proposé un cadre légèrement adapté, basé sur l'hypothèse que les sources sont des séquences qui ont été copiées et ont divergé dans les génomes, formant ainsi des familles identifiables. Ainsi nous partons d'une famille de séquences S, et recherchons un ensemble minimal de sous-séquences sources D (les domaines) telle que chaque élément de S peut être exprimé comme la concaténation d'éléments de D ayant évolué [214]. Du fait que la segmentation prenne en compte l'ordre de la séquence et plusieurs séquences en parallèle, on espère qu'elles produisent une partition plus fine que celle basée sur une analyse en composition d'une seule séquence.
Plutôt que de se focaliser sur le nombre maximum de domaines, on suppose que les domaines ont une taille minimale $m = MinSizeDomain$, qui est suffisante pour les caractériser de manière non ambiguë. Plus précisément, on suppose que chaque domaine est caractérisé par un mot w de taille au moins m et que chacune de ses occurrences correspond à un mot qui diffère de w par au plus $MaxErrors$ erreurs. Une erreur est soit une substitution, soit une insertion ou une suppression à une position donnée dans l'alignement multiple des séquences. Il se peut qu'aucune occurrence ne s'apparie exactement avec w.

Nous avons proposé un algorithme [214], qui part d'un alignement multiple des séquences à segmenter, puis recherche des points de coupures à partir d'une fonction d'estimation de l'hétérogénéité de la séquence. On effectue un alignement multiple en ne pénalisant pas l'insertion de brèches, ce qui décale naturellement les différents groupes de séquences, afin de limiter les variations à une position donnée de cet alignement. On admet ainsi un certain niveau de sursegmentation, qui est compensé par des regroupements lors des phases finales de l'algorithme. La fonction d'hétérogénéité est basée sur deux indices, qui mesurent dans une fenêtre de taille m si un alignement change vis-à-vis du nombre de domaines qu'il contient. L'indice le plus important calcule le nombre de séquences non présentes dans l'alignement multiple, c'est-à-dire représentées par une brèche (gap). Ce nombre devrait être constant pour une même configuration de domaines. Le second indice calcule sur l'ensemble des séquences, le nombre maximum de positions dans l'alignement ne présentant ni la lettre la plus fréquente, ni une brèche. On suppose en effet qu'un seul domaine existe dans un alignement seulement si les variations de sa séquence restent minimes dans l'alignement ($MaxErrors$ erreurs). Les points de coupure correspondent aux extrema locaux de la fonction d'hétérogénéité tels que soit le nombre de séquences change, soit le seuil $MaxErrors$ est atteint.
Une fois fixés les points de coupure, on passe à la deuxième phase de caractérisation des domaines. Pour cela, on classe l'ensemble des séquences d'un segment d'alignement en procédant récursivement par découpage binaire entre les séquences respectant le consensus à $MaxErrors$ près et celles ne le respectant pas.
Chaque ensemble de sous-séquences ainsi formé est aligné et reçoit une caractérisation par un profil HMM (Profile Hidden Markov Model) [70], un modèle probabiliste associé à la représenta-

tion d'un alignement de séquences. C'est cette caractérisation, et non pas la segmentation initiale, qui va servir de base à la recherche des domaines.
On repère sur l'ensemble des séquences, l'ensemble des positions d'occurrence des modèles produits (nous avons utilisé le logiciel HMMER [72]). Comme les modèles sont détectés indépendamment, il peut y avoir des inclusions ou des chevauchements d'occurrences sur les séquences. Un cas d'inclusion est typique d'un élément génétique mobile ayant intégré un autre élément mobile. Un cas de chevauchement élevé est interprété comme la présence d'un domaine unique et les modèles HMM sont fusionnés. Un cas de chevauchement restreint pour un même modèle est interprété comme une copie en tandem. Enfin, les autres cas de chevauchements révèlent un motif commun entre domaines plutôt qu'un nouveau domaine encadré par deux domaines plus petits car il serait difficile d'interpréter biologiquement un tel schéma de dépendances. Nous conservons toutes les relations naturelles observées entre domaines et conservons donc l'ensemble des domaines détectés au travers des modèles HMM, plutôt que de générer une intersection de domaines à chaque fois qu'il y a chevauchement. Ceci conduirait en effet à un émiettement rapide de la séquence et à une diminution de la pertinence biologique des domaines.

La dernière étape de l'algorithme est une double minimisation, qui porte à la fois sur la distance entre les occurrences de domaines et sur le nombre de domaines. En d'autres termes, comme nous l'avons indiqué dans notre formalisation, il s'agit de couvrir un ensemble de séquences par un ensemble de mots appartenant à un dictionnaire de référence (les domaines). La couverture doit minimiser la taille de cet ensemble de mots ainsi que la portion non couverte des séquences, ou couverte de façon multiple (la fonction objectif est simplement la somme de la taille des portions non couvertes ou des portions chevauchantes et de la taille des domaines utilisés). Nous avons proposé deux méthodes qui donnent de bons résultats en pratique.
La première est une approche gloutonne qui procède séquentiellement sur la liste des séquences, en propageant à chaque fois les choix de domaines effectués. Sur chaque séquence, on recherche le découpage en domaine qui minimise la fonction objectif. Étant donnée l'importance des choix effectués sur la première séquence, on effectue en réalité une autre étape de minimisation sur l'ensemble des premières séquences possibles.
La seconde méthode, développée dans un travail en collaboration avec P. Veber et R. Andonov, utilise une représentation des séquences sous forme d'un graphe d'intervalle où les noeuds sont des positions d'occurrences de domaines et les arcs indiquent une succession acceptable de domaines. On utilise pour le codage du problème un modèle linéaire avec deux types de variables binaires, les variables représentant l'utilisation d'un domaine particulier, et les variables représentant le fait qu'une séquence donnée utilise un arc donné du graphe d'intervalles. Le modèle a été résolu de manière exacte sur une quinzaine de séquences par un solveur générique, CPLEX (Ilog) [221].

La figure 2.9 montre le résultat d'une application de l'algorithme sur une famille de transposons de la plante *A. thaliana*, AtRep.

Nous travaillons actuellement sur une caractérisation plus précise de la flexibilité dans le cadre des séquences biologiques.
Basiquement, *la flexibilité* exprime ce fait que les unités élémentaires peuvent être assemblées par paires à une distance contrainte. On peut définir une notion de répétition flexible maximale (RFM) récursivement comme une paire (L, R)de mots ou de RFM, séparés par un espacement contraint

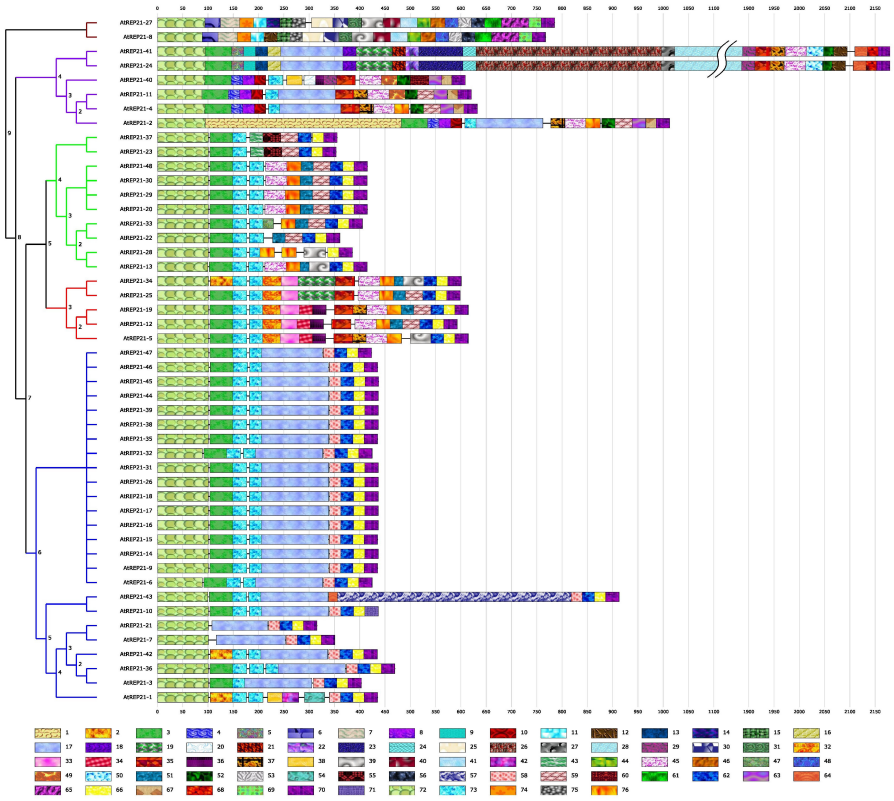


FIG. 2.9 – Décomposition en domaines et classification de la famille AtRep dans le génome de *A. thaliana*

X. La contrainte sur X doit dépendre de deux facteurs : sa taille par rapport à celle de L et R, et la variabilité de ses occurrences.

Nous avons concrètement travaillé sur une famille récente de transposons, les hélitrons, afin d'établir un recensement complet des éléments de cette famille [215]. On a pu ainsi proposer une nouvelle nomenclature pour cette famille, qui est actuellement reprise dans la base internationale RepBase [217]. La base de la caractérisation de ces hélitrons, comme pour d'autres éléments mobiles copiés dans le génome, repose sur une description adéquate de leurs extrémités. Une véritable combinatoire existe sur la composition des extrémités entre elles. Si l'on cherche à abstraire le problème, la notion de répétition flexible maximale semble un bon départ. Il reste ensuite à décrire le modèle complet d'un hélitron et sa dynamique d'insertion, ce qui suppose entre autres de caractériser les sites à l'intérieur de la séquence ainsi que les traces des lieux d'insertion. On passe alors du simple repérage d'éléments à la construction d'un véritable modèle, que l'on souhaite être opérationnel pour la recherche dans les banques de données.

Chapitre 3

Structures syntaxiques sur les génomes

Grammaire : vieille dame qui a toujours ses règles.
Définition de mots-croisés

La plupart du temps, analyser de manière linguistique une séquence biologique signifie pour leurs auteurs faire de la linguistique statistique, c'est-à-dire compter les occurrences de facteurs de taille donnée [30, 170, 171, 206]. Fondamentalement, on considère alors que toutes les séquences sont possibles et qu'un langage se réduit à la distribution de probabilités des facteurs au sein des séquences générées. Sans nier l'utilité d'une telle analyse, en particulier s'il s'agit uniquement de classifier les séquences, nombreux sont les auteurs qui perçoivent les limitations de cette approche qui ne permet pas de prendre en compte les corrélations fortes réduisant de fait les enchaînements possibles ou simplement fonctionnels de facteurs. Modéliser l'ensemble des mots qui peuvent être effectivement actifs dans la cellule, c'est se donner un formidable outil d'investigation scientifique des mécanismes du vivant. Un langage permet une très bonne abstraction du potentiel génétique, des structures spatiales ou de l'architecture fonctionnelle d'ensembles de séquences présentant des caractéristiques biologiques communes.

Très peu de travaux s'attaquent au problème de fond de l'existence d'une organisation complexe sur les éléments d'une séquence génomique. Cette architecture résulte de la superposition de contraintes multiples, dérivées du caractère actif de la molécule (phénomènes de régulation, de transposition,...), mais aussi de considérations générales sur les mécanismes cellulaires de gestion de la molécule (par exemple, enroulements autour d'histones) et sur le fait que son accessibilité doit être garantie dans certains contextes de transcription.

Notons d'emblée que nous n'aborderons pas du tout les aspects linguistiques liés au calcul naturel (DNA computing). Ce champ de recherche étudie des systèmes formels où la récriture est basée sur des opérations observées dans les systèmes biologiques (recombinaison, inversion, transposition ...). Une première liste de références est disponible dans [89] pour les lecteurs intéressés.
L'ambition de ce chapitre, au-delà du constat de la nécessité de recherches linguistiques plus sophistiquées, est de montrer quelques pistes et travaux qui dessinent un paysage beaucoup mieux formalisé de ce que pourrait être la modélisation sur les séquences biologiques. C'est aussi pour

l'informaticien l'occasion de rappeler que le paysage théorique des langages est loin de se réduire à la hiérarchie de Chomsky. Au niveau de l'enseignement comme de la recherche, nous avons besoin d'une vue beaucoup moins naïve des classes de langages, qui, comme les espèces, se coulent difficilement dans une organisation un peu trop cartésienne en un seul arbre.

3.1 De l'intérêt de la modélisation de séquences biologiques par des langages formels

A notre connaissance, une des premières mentions sérieuses de l'utilisation de langages formels pour décrire des structures dans les génomes date de 1984 [31].
L'article décrit, sur un exemple intéressant de phages à ARN, la possibilité de modéliser un langage incluant cette famille par une combinaison d'automates d'états finis, puis de modéliser la traduction de leurs gènes par des transducteurs.
Nous donnons sur la figure 3.1 la cartographie des gènes de MS2, un des membres de la famille des *Leviviridae*. Il s'agit de bactériophages à ARN simple brin à polarité positive s'attaquant principalement aux entérobactéries.

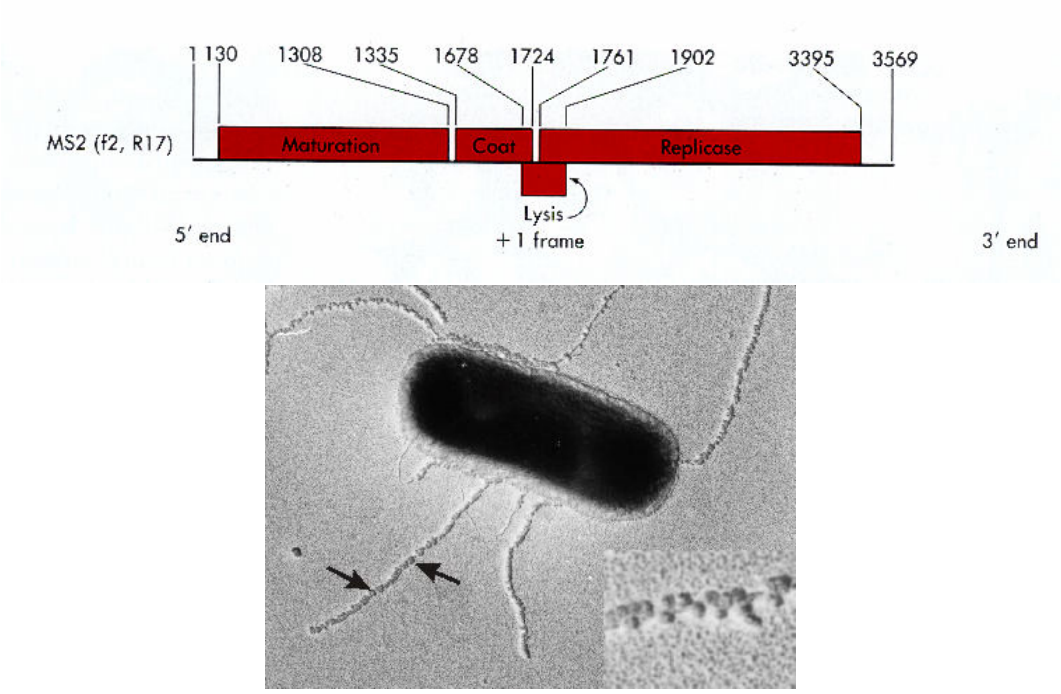


FIG. 3.1 – Carte génétique du virus MS2 et Accrochage de virus MS2 sur les flagelles de E. coli

Couramment utilisé dans les études en laboratoire pour étudier la résistance des virus dans l'environnement et l'efficacité des désinfectants, il s'agit du premier organisme pour lequel on a disposé du génome complet, en 1976 [80]. Il est essentiellement constitué comme la plupart des membres de la famille de 4 gènes dont l'un, celui de la lyse du virus, en chevauche deux autres :

celui codant pour l'enveloppe du virus (la capside) et celui codant pour la protéine synthétase de réplication.
La compacité est une caractéristique essentielle du code génétique : elle prend des formes multiples dont celle que nous venons de décrire de codages superposés par décalage de phase dans le cas de petits génomes. Plus généralement ce sont des réarrangements combinatoires de fractions de gènes que l'on observe. D'un point de vue linguistique, cette compacité correspond à une superposition de langages.

Si on s'intéresse au génome d'un point de vue statique, la résultante de cette superposition est le langage intersection de tous les langages correspondant. Brendel et Busse suggèrent que les langages réguliers sont très adaptés pour cette tâche [31], car la plupart des motifs caractéristiques sont simples (réguliers) et l'intersection et le produit d'un langage régulier sont encore des langages réguliers. Par exemple, la traduction de l'ARN du gène de capside en protéine de capside suppose une cascade d'événements : l'accrochage du ribosome sur le mot GGAG, la rencontre un peu plus loin d'un codon de démarrage AUG, une lecture d'une suite de triplets codant pour les acides aminés de la protéine, et enfin un triplet spécifique d'arrêt de la traduction (qui est un des mots UAA, UAG ou UGA). Plus formellement, si on pose $\Sigma = \{A, C, G, U\}$, $Triplets = \Sigma^3$, $Stop = \{UAA, UAG, UGA\}$ et $Codons$ = $Triplets$ - $Stop$, les séquences correctes doivent appartenir au langage

$$\Sigma^*.GGAG.\Sigma^*.AUG.Codons^+.Stop.\Sigma^*.$$

Le gène de lyse est en décalage +1 par rapport à ce gène. Si L_{lyse} est le langage régulier associé aux séquences possibles du gène de lyse, la séquence du génome doit également vérifier le langage intersection suivant :

$$\Sigma^*.GGAG.\Sigma^*.AUG.((Codons^+.Stop.\Sigma^*) \cup \Sigma.T^*.L_{lyse}) \cup \Sigma^*.$$

De proche en proche, on peut prendre en compte l'ensemble des contraintes connues sur l'enchaînement des gènes dans le génome du virus.

L'intérêt des langages réguliers semble particulièrement évident sur des familles de protéines. En effet, les sites actifs sur les protéines sont relativement bien localisés, et décrire une famille se ramène facilement à un problème de description de ces sites à l'aide d'une expression régulière. Les biologistes se contentent même en pratique d'un sous-ensemble très limité des langages réguliers en utilisant des expressions du type $\Sigma^*.E.\Sigma^*$, où E est un langage fini n'utilisant la disjonction que sur des lettres (syntaxe Prosite, [164]).
Un article récent dans Nature [144] et qui a eu un impact médiatique dans la presse américaine[1] montre que ce type élémentaire de modélisation linguistique est encore loin d'être acquis en biologie. Les auteurs y proposent la modélisation d'une classe particulièrement importante de protéines, qui ont des propriétés antimicrobiennes et antifongiques naturelles. Ce sont donc des molécules de choix à tester pour les laboratoires pharmaceutiques, face au développement de résistances aux antibiotiques classiques.

[1] Un article de S. Borenstein dans Discovery channel affichait par exemple "Bacteria, meet your new enemy : grammar" !

A partir d'un ensemble de motifs de type Prosite de taille $k = 10$ fixée, caractéristiques de la famille des peptides antimicrobiens, les auteurs de cet article construisent un langage fini intégrant les différents motifs. Ce langage est généré par toutes les possibilités de chevauchement, sur une longueur $k - 1$, de k motifs consécutifs. Cet artifice permet d'échapper à la restriction Prosite mais le langage aurait pu sans problème être décrit par une expression régulière. L'intérêt du travail réside surtout dans le test de l'activité antimicrobienne de nouveaux peptides appartenant à ce langage et différant fortement des peptides existants : pour 1/10 des peptides générés, cette activité est très significative, ce qui est un résultat marquant dans un domaine où le taux de succès est extrêmement faible. Il s'agit donc d'une démonstration claire de l'intérêt d'une modélisation linguistique, obtenue avec des concepts très simples du point de vue de la théorie des langages.

Nous avons nous-mêmes travaillé sur les peptides antimicrobiens avec C. Alland, G. Ranchy, F. Bourgeon et C. Pineau, dans le cadre d'une coopération avec un laboratoire de l'INSERM (Germh, dirigé par B. Jegou). Le but était légèrement différent puisqu'il s'agissait de trouver des peptides inconnus existant réellement dans les génomes. Ce travail a donné lieu à la thèse de F. Bourgeon, à un contrat avec la Société Innova Proteomics et au dépôt d'un brevet sur les peptides antimicrobiens générés.
Il s'agissait plus particulièrement, à partir de la connaissance des séquences de quelques protéines de la famille des $\beta-$défensines dans des génomes de vertébrés, de rechercher dans le génome humain de nouvelles protéines de cette famille. Quatre exemplaires de cette protéine étaient connus quand nous avons démarré l'étude. Nous avons construit un modèle caractéristique de cette famille par apprentissage automatique sur les séquences connues. Pour généraliser au mieux les exemples de séquences en un modèle qui soit caractéristique de la famille, nous avons tenu compte de la connaissance de familles proches aux propriétés différentes, les α-défensines et les chemiokines. La deuxième partie de ce chapitre abordera la question des méthodes utilisées pour construire de tels modèles. Nous nous contenterons d'indiquer ici que nous avons produit une expression de type Prosite, puis filtré et trié les mots acceptés à l'aide d'un score calculé à partir de statistiques sur les facteurs de ces mots. Cette recherche a conduit à la découverte d'une trentaine de nouvelles défensines qui n'avaient pas été détectées par l'approche, la plus performante actuellement, des modèles de Markov cachés.

Les langages réguliers sont aussi utiles sur l'ADN. Ainsi, Kangaroo [24], un simple outil de reconnaissance d'expressions régulières dans les génomes, est appliqué par ses auteurs sur un problème de reconnaissance de différents types de cancers colorectaux. Le problème est réduit à la recherche de mots de taille variable dans les gènes, constitués de la répétition d'un seul acide nucléique, qui provoquent plus facilement des mutations pathogènes.

Les outils ne manquent pas dans ce domaine de modélisation par langages réguliers. Citons parmi les outils publiquement disponibles les plus utilisés/récents FindPattern de GCG [59, 148](lignes de commande, reconnaissance d'expressions régulières, matrices de profils et expressions booléennes).

3.2 De l'intérêt d'aller au delà des langages réguliers pour modéliser les séquences

On peut dès à présent établir un certain nombre de remarques sur l'adéquation de la modélisation de structures génomiques par des langages réguliers :

- Brendel et Busse [31] concevaient uniquement les langages réguliers comme un outil mathématique pour formaliser les modèles de séquences génétiques. Cependant, c'est bien le niveau, beaucoup plus exigeant, du *langage informatique assistant opérationnellement le biologiste dans la construction puis la vérification des modèles* qui doit être développé.
  La question est, tout en minimisant l'introduction des concepts utiles, de fournir le bon niveau d'expressivité à ce langage. Par exemple, comment écrire simplement le fait que la séquence GGAG précédente est plus généralement une sous-chaîne "suffisante" de la séquence dite de Shine-Dalgarno AGGAGG, ou encore comment décrire correctement l'ordre chevauchant d'enchaînement des gènes avec des décalages éventuels de phase et des distances contraintes entre les différentes parties ? Tout ceci est réductible à un langage régulier si on ne s'intéresse qu'au problème de l'appartenance au langage, c'est à dire à la classification d'une séquence par rapport à différents modèles. Cependant, c'est oublier que l'intérêt d'un modèle réside autant dans son aspect explicatif que prédictif.
  La question n'est pas simplement celle du langage reconnu mais aussi de sa grammaire. Autrement dit, au-delà de l'appartenance, c'est le problème plus riche de l'analyse d'une séquence qui se pose, c'est-à-dire la manière dont une séquence est reconnue par un modèle qui est une grammaire, résumée dans son arbre d'analyse.
- Même dans le cas simple de génomes viraux, *les langages réguliers ont une expressivité insuffisante* pour traduire la conservation observée des structures secondaires pour certaines régions de la molécule d'ARN.
  Ainsi dans notre exemple, on observe dans la partie 3' du gène de la synthétase (positions 3349-3379) une structure de tige(hélice)-boucle conservée dans l'ensemble des Leviviridae (cf son consensus figure 3.2). Bien qu'une séquence consensus soit affichée, la conservation

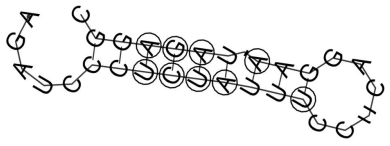


FIG. 3.2 – Structure secondaire consensus de Levivirus, positions 3349-3378 de MS2

porte bien sur la structure et non sur la séquence elle-même, ce qui est souligné dans la figure par des acides nucléiques entourés d'un cercle : pour chacun de ces acides, on peut exhiber dans au moins une espèce une mutation qui permet de conserver la structure alors que la séquence varie. La séquence correspondante de MS2 est

$$\mathbf{C}\mathit{G}\mathbf{G}\mathit{U}\mathbf{U}\mathit{CC}\mathbf{CAC}\mathit{AUU}\mathbf{C}\mathit{CCUCAGGA}\mathbf{G}\mathit{U}\mathbf{GUG}\mathit{GGC}$$

alors que la séquence consensus est

$$\mathbf{G}\mathit{G}\mathbf{A}\mathit{U}\mathbf{C}\mathit{CC}\mathbf{UCU}\mathit{AUU}\mathbf{U}\mathit{CCUCAGGA}\mathbf{A}\mathit{U}\mathbf{AGA}\mathit{GGC}.$$

Nous avons souligné en gras les caractères mutés. On constate de façon significative que les mutations n'apparaissent pas ponctuellement mais que par exemple le mot UCU, apparié avec son symétrique dyadique Watson-Crick AGA dans le consensus, devient CAC apparié symétriquement avec GUG pour le virus MS2. Cette conservation est probablement fonctionnelle, sans qu'on en connaisse actuellement le mécanisme. On remarquera que la modélisation de cette conservation est un langage algébrique. On sait d'autre part que les génomes exhibent des structures non algébriques, la plus connue étant la structure de pseudo-noeud dans l'ARN (cf figure 3.3). Cependant, la simple superposition de contraintes algébriques est susceptible de mener à cette complexité. En effet, la classe des langages algébriques n'est pas close par intersection.

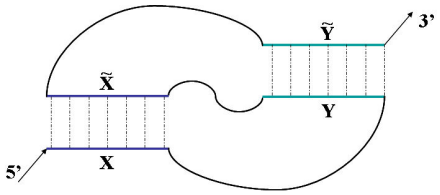


FIG. 3.3 – Structure secondaire de pseudo-noeud dans l'ARN

– Enfin, notons que si un génome peut être étudié d'un point de vue statique, on ne peut oublier qu'un organisme vivant est avant tout un système dynamique en interaction avec son environnement. Au niveau de la modélisation, ceci a des conséquences importantes. *On doit pouvoir obtenir des analyses concurrentes d'une même chaîne*, et en particulier permettre des grammaires ambiguës.
Dans notre exemple, la traduction des gènes du virus produira de manière exclusive soit le gène de lyse soit les gènes de capside et de réplication. Une manière simple de modéliser ce qui se passe au moment de la traduction est de produire deux solutions alternatives dans l'analyse. On sait aussi que certains ARN ou certaines protéines peuvent adopter en fonction du contexte des structures spatiales différentes pour une même séquence : c'est le cas bien connu du prion, protéine impliquée dans la maladie de la vache folle, ou le cas des atténuateurs pour l'ARN. Ces alternatives représentent un moyen essentiel de régulation pour le vivant.

D'une manière plus générale, l'ensemble des processus qui agissent sur les macromolécules biologiques et les métabolites (copie, transcription, épissage, traduction, régulation de transcription, interaction entre protéines...) peuvent être interprétés comme des transformations entre différentes chaînes. Cette dynamique de passage d'une séquence à l'autre fait pleinement partie des problèmes de modélisation sur les séquences génomiques. Brendel et Busse proposaient déjà d'utiliser les transducteurs pour modéliser la traduction d'un ARN en protéines.
Collado-Vides a également esquissé une approche par grammaires transformationnelles pour rendre compte des différents mécanismes de régulation de l'expression d'opérons [44]. Il propose dans un premier temps de construire sur la séquence du génome une "G-structure" reflétant l'enchaînement des zones promotrices, des protéines de répression ou d'activation

et des gènes de structure. Puis il propose de transformer cette G-structure par un ensemble de règles en "E-structure" fournissant une abstraction de la régulation de l'expression des gènes. On effectue pour cela des déplacements dans la E-structure correspondant aux différentes étapes de la régulation (intégration de l'inducteur dans le cytoplasme, accrochage de l'inducteur à la protéine de régulation, largage ou accrochage de la protéine sur le site de fixation de la molécule d'ADN). De tels travaux restent cependant très préliminaires et ne semblent pas avoir été poursuivis.

Face à ce constat de la nécessité d'un vrai langage de modélisation adapté à la description de structures non régulières, les outils spécifiques de recherche de motifs dans les séquences se sont multipliés.
La modélisation des ARN a été en particulier bien étudiée, en s'appuyant sur des grammaires définies a priori. Ainsi, Palingol [25] est un petit langage de programmation spécialement développé pour l'analyse des ARN, qui suppose que les structures sont disponibles au niveau lexical (les constituants sont les tiges-boucles de la séquence à analyser), et permet d'écrire des contraintes sur la taille des différents constituants, leur composition, la distance et les relations entre eux. Il est proche dans l'esprit des langages de contraintes, thème que nous développons par la suite, mais reste un langage de programmation spécifique au domaine.
Le développement de grammaires algébriques pour la reconnaissance de structures d'ARN pose un ensemble de questions fondamentales par rapport à la modélisation par des langages. Considérons la grammaire la plus évidente pour ces structures :

$$G = (\{S\}, \Sigma = \{a, u, g, c\}, S, P), \text{où } \tilde{x} \text{ est un acide apparié à } x \text{ et}$$
$$P = \{S \to xS,\ S \to Sx,\ S \to xS\tilde{x},\ S \to SS,\ S \to \epsilon \mid x \in \Sigma\}$$

Cette grammaire est bien sûr ambiguë. En fait, il existe de très nombreuses analyses possibles pour une même séquence d'ARN et ceci pose un réel problème du point de vue de la modélisation. Le biologiste recherche la structure que va réellement adopter la molécule et souhaite donc non seulement une analyse possible mais l'analyse la plus probable.
Bien que nous n'abordions pas ce problème dans ce document, il a des répercussions sur les propriétés souhaitables des grammaires. En effet, une bonne façon de probabiliser les productions d'une grammaire (on parle alors de grammaire stochastique) est d'associer à chaque règle une probabilité (qui peut éventuellement être décomposée en une probabilité de transition entre non-terminaux d'un côté et une probabilité d'émission de nouveaux non-terminaux de l'autre), en s'assurant que la somme des probabilités est à 1 pour l'ensemble des règles avec un même non-terminal à gauche. L'ambiguïté est un facteur de complexité pour l'estimation de la probabilité d'une structure par apprentissage sur un ensemble de séquences dont la structure spatiale a été résolue : il faut en effet parcourir pour cela l'ensemble des analyses possibles de la séquence cible, compatibles avec la structure cible.
En réalité, on a besoin d'un critère relâché d'ambiguïté structurelle, en termes d'équivalence des structures spatiales. Il s'agit ici de faire en sorte que la grammaire ne produise qu'une seule analyse d'une séquence avec des appariements fixés. L'ambiguïté d'une grammaire algébrique est un problème indécidable dans sa généralité mais l'ambiguïté structurelle peut être menée pour une grammaire et des séquences particulières. L'article [64] montre très clairement la méthodologie de

conception possible de ces grammaires et l'influence des choix effectués sur la qualité de prédiction de séquences réelles. Une des meilleures grammaires obtenues du point de vue de son pouvoir prédictif est la suivante :

$$G_1 = (\{S, L, R\}, \Sigma = \{a, u, g, c\}, S, P), \text{où } \tilde{x} \text{ est un acide apparié à } x \text{ et}$$
$$P = \{S \rightarrow LS,\ S \rightarrow L,\ L \rightarrow xR\tilde{x},\ L \rightarrow x,\ R \rightarrow xR\tilde{x},\ R \rightarrow LS \mid x \in \Sigma\}$$

Il est possible d'écrire des grammaires beaucoup plus petites, mais qui donneront des résultats très insatisfaisants en prédiction :

$$G_2 = (\{S\}, \Sigma = \{a, u, g, c\}, S, P), \text{où } \tilde{x} \text{ est un acide apparié à } x \text{ et}$$
$$P = \{S \rightarrow xS,\ S \rightarrow xS\tilde{x}S,\ S \rightarrow \epsilon \mid x \in \Sigma\}$$

Enfin, il est possible d'écrire des grammaires plus complexes, qui permettent de prendre en compte des effets contextuels limités :

$$G_3 = (\{S, L\} \bigcup \{R_x, x \in \Sigma\}\}, \Sigma = \{a, u, g, c\}, S, P), \text{où } \tilde{x} \text{ est un acide apparié à } x \text{ et}$$
$$P = \{S \rightarrow LS,\ S \rightarrow L,\ L \rightarrow xR_x\tilde{x},\ L \rightarrow x,\ R_y \rightarrow xR_x\tilde{x},\ R_x \rightarrow LS \mid x \in \Sigma\}$$

L'idée est à la fois de rapprocher la structure de l'arbre d'analyse de la structure spatiale réelle, et d'ajouter une sémantique aux règles en accord avec les connaissances que l'on veut inclure. Le problème général est celui de la projection d'une structure sur un arbre, avec la possibilité de conserver une information de taille bornée au niveau des noeuds. L'enjeu est également de pouvoir comparer de manière normalisée des structures entre elles et donc de faire de l'alignement multiple [65]. Une conclusion importante de l'étude est qu'*o*n peut obtenir avec des modèles grammaticaux bien conçus une prédiction comparable à celle des meilleurs algorithmes basés sur un calcul complexe de minimisation d'énergie. Ceci s'effectue avec beaucoup moins de paramètres, ce qui est un avantage certain du point de vue du réglage du modèle par apprentissage automatique, mais également en termes de connaissances, du point de vue de l'abstraction obtenue du modèle lui-même.

3.3 Variables de chaîne

Les travaux majeurs sur la modélisation de séquences génomiques par des langages formels sont dûs à Searls, qui a publié un grand nombre d'articles sur le sujet [195, 196, 197, 204, 198, 199, 201, 202, 203] et jeté les bases d'une étude sérieuse du domaine. Nous recommandons en particulier la lecture de [201], qui offre une bonne synthèse de ses contributions, tant sur le plan théorique que pratique. Searls ne s'est en effet pas contenté d'une simple formalisation des structures rencontrées dans les séquences biologiques dans le cadre de la théorie des langages. Il a également supervisé des développements qui ont permis pour la première fois de construire des grammaires pour l'analyse de données génomiques réelles [204, 63].
A la base des réflexions de Searls, il y a la difficulté de traduire les phénomènes biologiques fondamentaux à l'aide de grammaires algébriques ou sensibles au contexte. En particulier la copie, inversée ou non, qui est une structure non algébrique à la base de l'évolution des génomes, est délicate et très peu naturelle à modéliser. En balance avec cette nécessité d'étendre le cadre classique,

le but de Searls est de s'éloigner au minimum de l'excellent compromis expressivité/efficacité dont bénéficient les grammaires algébriques. Il rejoint en cela les résultats obtenus par la communauté d'analyse du langage naturel, qui a développé pour ses besoins propres ce qu'A. Joshi nomme des grammaires "midly context sensitive" [119].

Searls a approfondi une idée a posteriori simple, celle d'introduire **un type abstrait "variable de chaîne "** dans les grammaires formelles, avec la volonté claire d'aboutir à un langage de modélisation adapté aux séquences génétiques. La notion de variable de chaîne elle-même n'était pas nouvelle : les variables ont été introduites par D. Angluin dans les langages de patterns [12] et, bien qu'elle soit peu utilisée, on retrouve une notion de variable appelée "référence arrière" ("backreferences") [6] dans la plupart des outils pratiques de recherche d'expressions régulières (egrep, awk, sed, perl, python, JavaScript, JScript,...).
Les langages de pattern ont été bien étudiés, en particulier dans le domaine de l'apprentissage automatique du fait de l'appartenance de D. Angluin à cette communauté. Au niveau théorique, nous conseillons la lecture des travaux de K. Salomaa [192]. Les langages de pattern sont définis par des expressions qui sont des mots finis sur un alphabet terminal fini augmenté d'un ensemble réduit ou infini de variables. Les variables de chaîne sont des variables mathématiques, qui doivent représenter (s'instancier uniformément avec) un même mot dans toutes ses occurrences. On exige en général qu'une variable ne puisse être instanciée par le mot vide (NE ou "non erasing patterns" ; dans le cas contraire on note E la classe des patterns dont les variables peuvent s'effacer). E et NE sont incluses dans la classe des langages sensibles au contexte, incomparables avec la classe des langages réguliers et algébriques, non closes pour la plupart des opérations hormis la concaténation et l'inversion. L'analyse de l'appartenance d'un mot à un pattern est un problème NP-complet.

La notion de référence arrière est un cas particulier de **grammaire à dérivation contrôlée**, où l'on restreint la manière dont les règles de la grammaire peuvent être enchaînées. On trouvera dans [167] un exposé général et concis sur ces expressions régulières étendues nommées expressions régulières synchronisées (SRE), qui signale sans l'approfondir la possibilité d'utiliser ces expressions pour l'analyse des séquences génétiques. Signalons une forme très réduite de SRE disponible dans un outil récent d'analyse de séquences d'ADN, Pattern Locator [155]. On peut uniquement y faire référence à une position donnée dans le motif (p. ex. $'\#NNN(-3)(-2)(-1)'$ est la syntaxe décrivant une tige de taille 3).
Dans les SRE, on peut introduire deux types de variables : les variables de chaîne, à condition qu'elles aient été déclarées par une sous-expression délimitant le langage de ses substitutions possibles, et les variables de puissance, substituables par un entier. Ainsi, l'expression $a^x.(\Sigma^*) : y.y.a^x$ reconnaît le langage $\{a^n.w.w.a^n, n \in \mathbb{N}, w \in \Sigma^*\}$. On constate que les langages de pattern sont un cas particulier de ces SRE. La classe des SRE contient par définition la classe des langages réguliers mais est incomparable avec celle des langages algébriques[2].
Si l'on se restreint aux expressions où les variables ne sont pas imbriquées (1-SRE), on démontre facilement l'inclusion de cette classe dans une sous-classe de grammaire à dérivation contrôlée, les "scattered context grammars sans règles d'effacement". Ces dernières constituent simplement une extension parallèle des grammaires algébriques. Les règles de production y sont regroupées

[2]assez curieusement, les langages de copie sont des SRE mais les palindromes n'en sont pas !

en sous-ensembles de règles devant être dérivées de manière parallèle. Cette synchronisation des règles permet effectivement de générer plusieurs fois un même contenu (variable de chaîne) ou de boucler un même nombre de fois sur une application récursive de règle (variable de puissance).
Bien sûr, l'analyse de l'appartenance d'un mot à un SRE est un problème NP-complet, et ceci même si on ne met que des variables de puissance. Il existe en fait un résultat plus précis, intéressant pour la modélisation dans les séquences biologiques, qui établit la complexité de l'analyse d'une expression avec k variables sur une séquence de longueur n comme un polynôme en n^k.

La notion de variable apparaît également de façon fondamentale dans le cadre de la programmation logique sur lequel s'appuie Searls, cadre dont le développement est d'ailleurs dû à d'autres problèmes d'analyse de séquences naturelles : les textes [45, 42, 46]. La notion de chaîne est étendue à des listes de termes quelconques et comme en Lisp, on ne peut travailler dans ce contexte qu'avec une variable, en queue de liste. La programmation logique a ensuite pris le virage de la programmation par contrainte, où on généralise la notion d'unification par la notion de résolution d'un système de contraintes. Dans Prolog III puis dans Prolog IV, on permet ainsi d'introduire de façon plus souple les variables de chaîne, à la condition expresse que le système puisse inférer simplement la longueur des chaînes [157, 21].
Ainsi, un palindrome biologique s'écrira facilement avec une expression du type :

$$palindrome(Y) : -(Y =' c'.X.'g'; Y =' a'.X.'t'; Y ='''), palindrome(X).$$

car la taille de X peut être immédiatement déduite de celle de Y. Par contre la copie reste inaccessible car une expression $X.X$ contient 2 variables de taille inconnue et il ne peut y avoir syntaxiquement qu'une seule liste de taille inconnue.

Pour **traiter les chaînes dans un système de résolution de contraintes**, il faut introduire un système équationnel puis disposer d'un algorithme de résolution des équations sur des expressions syntaxiques. On s'intéresse pour cela tout particulièrement aux patterns, au sens où nous l'avons défini précédemment d'une séquence de lettres et de variables.
Unifier des patterns est un problème beaucoup plus difficile que l'unification classique des termes car il n'existe pas un unique plus grand unificateur. Il peut même en exister une infinité.
On dispose actuellement de deux algorithmes pour résoudre les équations sur les patterns, celui de J. Jaffar [114] et celui de W. Plandowski [176]. L'algorithme de J. Jaffar se base sur les travaux fondateurs pour le problème de la satisfiabilité de Makanin [61][3] et génère un ensemble d'unificateurs U complet (toute solution est une instance d'un élément de U) et minimal (aucun élément de U n'est une instance d'un autre). L'algorithme ne peut terminer que si U est fini. W. Plandowski relâche la contrainte de minimalité et obtient un algorithme plus simple (avec une complexité DEXPTIME quand même au lieu de EXPSPACE), qui finit toujours en produisant un multi-graphe fini représentant l'ensemble des solutions. Ces travaux sont relativement récents et mènent à des algorithmes de complexité qui reste élevée. Pour cette raison, ils n'ont pas été diffusés dans leur généralité dans les résolveurs de contraintes actuels. Notons que W. Plandowski a

[3]Cet article d'une cinquantaine de pages étend et présente de manière plus claire l'article d'origine qui est très difficile à lire. La preuve de correction de l'algorithme de satisfiabilité est une curiosité car dit-on une des plus complexes existant sur la correction d'un algorithme. Je ne m'y suis pas attaqué...

obtenu des complexités en o(n) et en o(n^6) pour les cas à 1 et 2 variables respectivement [108, 55] qui mériteraient d'être exploités.

Parmi les meilleures tentatives, A. Rajasekar propose un type restreint de variable de chaîne dans un système de programmation logique avec contraintes, appelé $CLP(S)$ [183].
Une variable de chaîne est ici un couple, $W : t$, où t est un entier positif et où W peut s'unifier avec une séquence de taille t. Une théorie équationnelle sur les patterns avec des variables de chaînes de taille connue est beaucoup plus simple car on retrouve une notion d'unificateur le plus général unique, ce qui permet à A. Rajasekar de proposer la concaténation de variables arbitrairement, avec des contraintes qui portent soit sur le contenu soit sur la taille des chaînes, en résolvant d'abord les contraintes sur les tailles. L'algorithme de réduction peut par exemple résoudre une équation comme

$$X : 2.Y : 3 \;=\; Y : 3.X : 2$$

en fournissant la solution

$$X \;=\; Z : 1.Z : 1, \;\; Y \;=\; Z : 1.Z : 1.Z : 1.$$

L'auteur a travaillé ensuite dans le cadre des bases de données relationnelles en utilisant ces mêmes variables comme attributs [184]. Un prototype a été développé, S-log, en Prolog et C. Le langage de requêtes a été conçu pour traiter différentes applications dont les séquences génétiques : il permet d'exprimer la concaténation, la disjonction et la puissance de chaîne. Egalement, l'utilisateur peut écrire ses propres fonctions sur les chaînes (par exemple générer le brin complémentaire d'une séquence d'ADN) et dispose de deux fonctions prédéfinies : $\#(N_1, N_2, e)$, qui filtre la présence de e dans les chaînes dont la taille est comprise entre N_1 et N_2, et $@(i, d, m, s, e)$, qui produit les chaînes avec au plus i insertions, d délétions, m mutations et s échanges de caractères par rapport aux chaînes de e.

Depuis les grammaires attribuées [9], on dispose d'un mécanisme largement repris en programmation logique (Definite Clauses Grammars, Gapping Grammars...) : faire apparaître les variables pas simplement en partie droite de règle (éventuellement en plusieurs exemplaires s'il s'agit de modéliser des phénomènes de copie), mais également en partie gauche comme argument de non terminal. De cette façon, on peut effectuer un passage de paramètres de l'application d'une règle à l'autre. Searls autorise simplement le passage de variable, seule possibilité de sensibilité au contexte que permet son formalisme, afin de rester au plus près des grammaires algébriques. Notons qu'il s'éloigne en cela des développements réalisés en analyse du langage naturel, où les chercheurs ont insisté sur des mécanismes sensibles au contexte pour traduire les mouvements des composants dans les phrases.

Searls distingue également deux alphabets, celui de spécification et celui de la chaîne analysée, afin de rendre compte des différents types de transformation agissant sur les séquences biologiques. Il introduit ces transformations via des substitutions orientées (sens direct ou inverse) permettant de passer d'un alphabet aux ensembles de parties de l'autre.
De manière plus formelle, les SVG (String Variable Grammars) sont définies comme suit :

définition 8. Grammaire à variable de chaîne (SVG) Une *Grammaire à variable de chaîne* est un octuple (Δ, Γ, Σ, N, V, S, F, P) où :

- Δ, Γ et Σ sont respectivement les alphabets des spécifications, des fonctions et des séquences ;
- N et V sont respectivement les ensembles de non-terminaux et de variables ;
- S est l'axiome initial de la grammaire ;
- F est un ensemble de substitutions finies de Δ dans 2^{Σ} associées aux symboles de Γ. Les substitutions sont étendues classiquement des lettres aux mots et signées : pour toute substitution f, et tout mot aW de Δ^+, si '.' dénote un produit d'ensembles, on a $f^+(aW) = f(a).f^+(W)$ et $f^-(aW) = f^-(W).f(a)$;
- P est un ensemble de règles de production de type $A \rightarrow \Phi$ ou $B(X) \rightarrow \Phi$, avec $A \in N$, $B \in N - \{S\}$, $X \in V$ et $\Phi \in (\Sigma \cup N \cup (V \times \{+,-\} \times \Gamma) \cup (N \times V))^*$

Du point de vue de la hiérarchie de Chomsky, Searls a montré que les SVG contiennent les grammaires algébriques et sont incluses strictement dans les grammaires indexées. Elles correspondent ainsi à une nouvelle classe de grammaires "midly context sensitive" puisqu'elles se placent de façon identique dans la hiérarchie des langages et diffèrent strictement des Tree Adjoining Grammars de Joshi d'une part[4] et des Reduplication PushDown Automata[5] d'autre part.

3.4 Les analyseurs en pratique

Au-delà de cette formalisation, quelques auteurs ont proposé un analyseur pour les séquences génétiques dans le cadre de la programmation logique. A notre connaissance, aucun de ces analyseurs n'est actuellement disponible[6]. Searls a très certainement été le plus loin dans cette voie. Il a proposé un langage et un analyseur pratiques, Genlang, qui introduisent un certain nombre de concepts supplémentaires par rapport au cadre formel des SVG.
Par exemple, curieusement, le formalisme SVG n'autorise qu'une seule variable dans la partie gauche d'une règle, alors que Genlang en permet plusieurs. L'ajouter ne pose pas de problèmes théoriques, la classe des SVG restant alors incluse dans la classe des grammaires indexées. En effet, la transformation SVG-grammaires indexées s'effectue en gérant dans la pile d'index la déclaration des variables de chaînes rencontrées avec leurs valeurs associées. Gérer plus de variables en partie gauche consiste juste à initialiser la pile avec plus de valeurs.
Par contre, autoriser une seule variable restreint inutilement le langage. Pour illustrer ceci, il nous suffit de reprendre l'exemple des CRISPRs abordé au précédent chapitre. De manière abstraite légèrement simplifiée, un modèle de CRISPR s'écrit comme un enchaînement d'un mot leader, d'une série de répétitions séparées par une distance fixe, et du complément inverse du mot leader. En syntaxe SVG augmentée pour autoriser plusieurs non terminaux à gauche, cela s'écrit :

$$crispr \rightarrow Leader, Unite, Insert, serie(Unite, Insert), Leader^{-wc}$$
$$serie(Unite, Insert) \rightarrow Unite, Insert^{+atgc}, serie(Unite, Insert)$$
$$serie(Unite, Insert) \rightarrow Unite, Insert^{+atgc}$$

[4]Le langage des www où w est un mot quelconque peut être décrit par des SVG mais pas des TAG.

[5]Le langage $a^n c^n g^n, n \geq 1$ peut être décrit par des SVG mais pas des RPDA.

[6]Après une recherche sur le web, nous avons contacté l'ensemble des auteurs concernés. David Searls et Carlsten Helgesen n'ont plus de copie de leurs programmes et les autres ne nous ont pas répondu. Cela pose un problème clair de maintien des apports scientifiques en ingénierie.

avec les non-terminaux commençant par une minuscule, les variables de chaîne commençant par une majuscule, et la définition des fonctions de substitution comme suit.

$$wc : a \rightarrow \{t\}; t \rightarrow \{a\}; g \rightarrow \{c\}; c \rightarrow \{g\}.$$
$$atgc : a \rightarrow \{a,t,g,c\}; t \rightarrow \{a,t,g,c\}; g \rightarrow \{a,t,g,c\}; c \rightarrow \{a,t,g,c\}.$$

A l'inverse, la notion de substitution, si fondamentale dans l'expressivité des SVG, n'est disponible que de manière restreinte dans Genlang. La complémentation inverse est proposée dans le langage ($\tilde{X}$ est la notation du complément inverse de X) mais les spécifications ne peuvent se faire que de manière indirecte via la construction des arbres d'analyse.
Globalement, Genlang peut être vu comme un laboratoire d'idées pour un ensemble de concepts qui semblent pertinents pour l'écriture de modèles réalistes sur les séquences biologiques. Les points suivants me semblent les plus importants :

- A tout composant peut être attaché un *ensemble d'attributs liés à son analyse*. Le système gère ainsi le coût de l'analyse (cost), la position de la chaîne analysée (span), sa taille (size) et la chaîne elle-même (list). Les composants des parties droites des grammaires peuvent être ainsi associés à des contraintes sur ces attributs. Les parties gauches peuvent construire l'arbre d'analyse via un attribut parse et un attribut label.
- Il existe également une *variable de position*, qui peut être manipulée via l'opérateur @. Ainsi $@X$, où X est une variable, unifie X avec la valeur courante de position dans l'analyse de la séquence. Une fois la variable unifiée, les autres occurrences de la variable permettent d'établir des contraintes de position. Ainsi $@X + 1?$ et $@end?$ vérifient respectivement que l'analyse est positionnée en $X + 1$ et que la chaîne a été totalement analysée. Il existe également un positionnement absolu sur la chaîne d'analyse : $@100$, $@X + 2$ sont des relations autorisées. Une autre manière de spécifier le positionnement est l'utilisation de distances, les *gaps*, correspondant à des chaînes quelconques de taille comprise entre deux bornes pouvant aller de 0 à l'infini.
- *La stratégie d'analyse peut être partiellement contrôlée* via un ensemble de primitives assez complexes. On utilise pour cela des non-terminaux spécifiques qui se dérivent en la chaîne vide avec un effet de bord sur l'analyse. $@once$ est l'équivalent d'un cut prolog, qui arrête la recherche de solutions alternatives pour la partie de règle à gauche de ce symbole. $@stay$ est une primitive qui permet de procéder à une analyse en regard avant (lookahead) : une règle $S \rightarrow A, @stay, B.$, provoque l'analyse de A puis de B et en cas de succès, le résultat de l'analyse est seulement celui de A. En partie gauche de règle, il existe des paramètres pour régler une stratégie en profondeur, en largeur ou meilleur d'abord. On peut également utiliser le paramètre $once$, qui signifie qu'il y a au plus une solution à produire ou le paramètre $skip$ qui est une forme originale de retour arrière indiquant que toutes les solutions sont non chevauchantes.
- Enfin, il est possible d'inclure l'exécution de n'importe quel code Prolog à l'intérieur d'une règle, ce qui bien sûr fait échapper totalement le système aux limitations des SVGs et permet en pratique de mettre en oeuvre des contraintes et le calcul de statistiques.

Parmi les travaux proches de Genlang, on citera BGG (Basic Gene Grammars) [140], qui présente peu d'innovations sur le plan langage. On retrouve ainsi des variables de chaîne simples (pas

de substitutions), des gaps, des coûts programmables et optimisables sur lesquels on peut exprimer des contraintes et des variables de positionnement (@1) pour l'appariement partiel. L'article explique clairement comment mettre en oeuvre le "chart parsing" dans ce cadre pour diminuer la complexité de l'analyse [7] et introduit deux notions dans le langage pour en tirer profit : une notion d'analyses recouvrantes ou contiguës, et une priorité gauche ou droite de l'ordre d'analyse. Les auteurs montrent également sur une application de reconnaissance de promoteurs chez Escherichia coli la souplesse et l'intérêt de tels langages pour intégrer différentes connaissances issues de la littérature ou de programmes d'apprentissage.

PALM [103] est un concurrent direct de Genlang, mais n'a été développé que sous forme d'un prototype en Prolog ne disposant pas d'outil de contrôle de la stratégie. Au niveau du langage, quelques ajouts intéressants sont proposés mais non discutés du point de vue complexité : la répétition d'un motif, l'intersection de motifs et une forme limitée de négation [8]. Comme dans le cas de Genlang, les auteurs introduisent des attachements procéduraux qui permettent en particulier de traiter des calculs statistiques [9].

Un autre analyseur général a été développé par rapport au langage proposé par A. Brazma et D. Gilbert [29], avec une version en CLP(FD) et une en formalisme CSP développée en C++ [74]. Il ne semble pas cependant que le système ait dépassé le stade de démonstration et ait été appliqué sur des données conséquentes. L'implémentation des variables de chaîne passe par des listes de taille bornée de couples (lettre,position) qui semble assez gourmande en pratique. Le langage lui-même n'offre pas de nouveautés par rapport à Genlang. On y trouve des contraintes sur la longueur (avec recherche d'optimum possible) sur le contenu et sur la position de début ou de fin des chaînes, sur la distance entre chaînes et sur les relations entre chaînes. Ces dernières peuvent prendre la forme d'une relation symétrique sur les lettres ou d'un morphisme lettre à lettre orienté dans le sens direct ou inverse. Une notion de coût de l'analyse avec appariement approché est également disponible.

3.5 Logol, un langage de modélisation sur des séquences biologiques

Dans la lignée des travaux de Searls et en essayant de rationaliser ses apports tant sur le plan théorique que pratique, nous développons avec C. Belleannée une plate-forme ambitieuse pour la recherche de motifs complexes sur des séquences nucléiques ou d'acides aminés.
Le langage, appelé Logol, autorise l'écriture d'une forme particulière de grammaire à clauses définies, fondée sur la distinction de deux niveaux de séquences [20]. Ces deux niveaux, *le niveau abstrait du modèle et le niveau concret de l'occurrence*, lui donnent toute sa cohérence et une expressivité un peu différente de celle de Genlang. Le résultat d'analyse est une structure qui reflète

[7] Genlang implémente aussi un mécanisme de tabulation de mots de taille fixée.

[8] $\{m_1, \ldots, m_n\}$ signifie "tous les mots qui ont une taille comprise entre celle de la plus petite et de la plus grande chaîne reconnue par les motifs $m_1, \ldots, m_n$ en dehors de ceux reconnus par ces motifs.

[9] De façon intéressante, les auteurs illustrent l'intérêt de ces comptages sur la traduction de motifs purement descriptifs de la banque de données Prosite qui ont été supprimés depuis ou remplacés par des profils. Ils ne rentraient pas dans la syntaxe régulière Prosite !

l'appariement d'une chaîne sur la structure d'un modèle.
Il ne suffit cependant pas de fournir un langage pour qu'il soit largement utilisé par la communauté des biologistes impliqués dans ces efforts de modélisation. Pour la plupart, accéder directement à un langage de programmation, même de haut niveau, représente un investissement trop important pour qu'ils deviennent des utilisateurs autonomes. Le projet complet inclut un langage graphique, ModelDesigner, pour aider le biologiste à écrire ses modèles et interpréter les résultats d'analyse en accord avec l'intuition spatiale des structures. Il est basé sur une architecture client-serveur développée par P. Durand. Nous nous concentrons cependant uniquement ici sur le coeur informatique, la spécification du langage Logol.

Formellement, Logol est un langage de contraintes sur les chaînes qui prend en compte deux niveaux de représentation sur celles-ci. Le premier niveau se réfère aux chaînes abstraites, que nous appelons *mots*, qui correspondent à un modèle idéal, pas forcément présent dans les séquences observées. Le second niveau désigne les chaînes concrètes, que nous appelons simplement *chaînes*, qui correspondent à la succession des bases observées dans les séquences biologiques. On retrouve ici une notion proche de celle suggérée par Searls, qui distingue un alphabet de spécifications et un alphabet de séquences ou celui des transducteurs qui distinguent un langage d'entrée et un langage de sortie. Nous pensons cependant que décrire le modèle et décrire la relation entre le modèle et ses instances sont des actions de nature différente qui nécessitent des mécanismes formels différents.

Nous nous proposons maintenant d'étudier plus en profondeur quelques aspects caractéristiques des séquences biologiques qui peuvent expliquer pourquoi les langages formels n'ont pas encore été largement diffusés et employés pour la modélisation sur les séquences génomiques. À partir de cette analyse des forces et des faiblesses des propositions actuelles, nous introduisons ensuite ce nouveau langage, Logol, qui se veut un pas supplémentaire en direction d'un langage de modélisation à part entière. On trouvera des exemples caractéristiques de la syntaxe du langage dans la table 3.5.

Duplication de séquences génomiques

Les séquences génomiques évoluent à travers un processus fondamental de duplication, éventuellement avec quelques erreurs, suivi par des mutations. La notion de variable de chaîne capture le fait que certains facteurs puissent apparaître plusieurs fois dans une séquence. Dans tous les langages que nous avons décrits jusqu'à présent, les variations possibles entre différentes occurrences d'une variable de chaînes sont prises en compte en introduisant une notion de coût. Plus précisément, en considérant la chaîne correspondant à la première occurrence, il est possible de contraindre les autres chaînes à être à une distance maximale fixée de celle-ci. Le processus n'est pas du tout symétrique car une des chaînes sert de référence aux autres.
Or en biologie la référence n'est pas toujours disponible ! En effet, ce que nous observons dans un ensemble de séquences biologiques d'une même famille dérive en général d'un ancêtre commun mais celui-ci n'est plus observable. Les multiples copies ont chacune leurs propres divergences par rapport au modèle ancestral. La difficulté de modéliser et de détecter de telles répétitions vient

du fait que la séquence référante ancêtre peut fort bien ne pas apparaître du tout dans la séquence. Par exemple, `AATT` peut être une séquence observée constituée de deux copies, `AA` et `TT`, d'un même ancêtre `TA` qui n'est pas observé. Ceci est typique des recopies observées dans les éléments génétiques mobiles. Il est tout simplement impossible de le formaliser avec les langages de patterns.
À l'inverse, Logol permet de mentionner et de se référer à un tel pattern abstrait, même s'il n'apparaît pas dans la séquence. Dans notre exemple, il suffit d'utiliser un modèle comme `X :1, X :1`. Ce modèle, qui utilise une variable de chaîne `X`, traduit le fait qu'un même mot `X` a été dérivé par deux mutations indépendantes.

Plus généralement, Logol considère deux niveaux pour les chaînes (c'est-à-dire les successions de bases) : un niveau qui désigne les chaînes abstraites -appelées *mots*- représentant des chaînes idéales qui n'existent pas nécessairement dans les séquences biologiques observées (`TA` dans notre exemple précédent), et un niveau qui désigne les sous-séquences concrètes -appelées *chaînes*- qui font partie des séquences observées (`AA` ou `TT` dans notre exemple).
Les modèles en Logol traitent d'ensembles de mots représentant les chaînes abstraites que l'on recherche, et l'ensemble de contraintes indiquant quelles sont les chaînes qui peuvent être acceptées comme des instances des mots, comme par exemple la localisation de la chaîne dans la séquence (on parle de *contraintes de chaîne*) ou les distorsions admissibles entre mots et chaînes (on parle de *contrainte de structures*).

D'un point de vue informatique, le problème de trouver une représentation commune d'un ensemble de mots est un problème classique d'apprentissage automatique (machine learning). Le système Winnower traite par exemple ce point [173]. Ce problème est aussi relié au problème de la chaîne centrale et médiane d'un ensemble de chaînes [159]. C'est un problème NP-complet et en pratique, seuls des cas bornés peuvent être résolus. C'est de manière générale le cas en biologie, où de multiples contraintes permettent de restreindre l'espace des chaînes admissibles. Par exemple, il existera un nombre fixé, généralement petit, de copies et la valeur de la distance maximale au centre pourra souvent être fixée (avec une correspondance relativement directe à l'âge de la copie).

Multi-motifs dans les séquences génomiques

Une autre caractéristique forte des séquences génomiques est qu'elles doivent maintenir des alternatives multiples dans leur code, cette forme de plasticité étant un atout de choix face à la nécessité des organismes de s'adapter à leur environnement.
Du fait de ressources finies, en particulier pour les petits organismes, ceci se traduit en une grande compacité de l'information codée.

Le recouvrement de gènes est observé dans tous les règnes du vivant, depuis les virus et bactéries où les gènes sont souvent très rapprochés jusqu'aux mammifères, incluant l'homme, au sein de régions riches en gènes [147]. Les traductions alternatives possibles de ces gènes partageant une portion de code sont probablement associées à des mécanismes importants de régulation.

Un cas tout à fait différent d'interprétations multiples d'un même code a été donné en exemple dans les travaux de D. Searls sur les ARN avec atténuateurs, qui correspondent à des mécanismes raffinés de contrôle de l'expression de gènes bactériens. Les atténuateurs contiennent une séquence qui peut s'apparier de deux manières différentes avec des séquences complémentaires pour former

dans chaque cas une structure de tige qui sera active ou non pour la terminaison de la transcription.

Nous pouvons enfin mentionner, comme facteur clé de la régulation de l'expression des gènes, la compétition qui s'établit entre sites de facteurs de transcription chevauchants.
Considérons par exemple le gène *eve* (even skipped) de la drosophile, bien connu car impliqué dans les stades précoces du développement embryonnaire et en particulier la formation de 7 bandes le long de l'axe antéro-postérieur. Chacune des bandes est régulée par une zone distincte du promoteur du gène. On trouvera dans [208] l'étude de la régulation associée à l'expression du segment numéro 2, en position [-931, -1601] par rapport au début de la transcription du gène.
Bicoid (*bcd*) et *hunchback* (*hb*) sont deux activateurs de transcription alors que *kruppel* (*Kr*) est un répresseur. Chacun de ces facteurs de transcription reconnaît un certain langage (motif) sur l'ADN et peut se lier à la zone promotrice de *eve* en des sites correspondant à ces motifs.
Les motifs reconnus par *bcd*, *hb* et *Kr* sont respectivement *SYWAATCC*, *WWTTTTWWG* et enfin *WYAAYCCDDY*, en utilisant l'alphabet IUPAC standard [10].
La séquence en [-1464, -1454] est *GTTAATCCGT*, et permet d'accrocher de façon chevauchante *bcd* et *Kr*.
La séquence en [-1031, -1015] est *ACTGGTTATTTTTTTG*, et permet d'accrocher de façon chevauchante *hb* et *Kr*.
Ce chevauchement syntaxique se traduit directement en potentiel de compétition entre activateurs et répresseurs au niveau de la régulation de l'expression.

Il est pratiquement impossible d'obtenir un modèle de tels mécanismes fondamentaux avec les langages disponibles. Avec Genlang, il est possible d'effectuer des retours arrière en un point donné de l'analyse de la séquence et effectuer ainsi des analyses multiples de motifs chevauchant, mais ceci au prix de descriptions lourdes comportant des éléments de stratégie d'analyse et de structures peu naturelles.

D'un point de vue formel, nous proposons de prendre en compte ces types d'ambiguïté à deux niveaux.
Au niveau le plus bas, Logol utilise deux symboles différents pour décrire la succession des entités dans une séquence. Le premier indique la succession de la fin d'une entité avec le début de la prochaine, selon la façon habituelle de décrire des mots consécutifs. Le second indique la contiguïté des positions de début de chaque entité. Ainsi, `"AAT","ATGG"` correspond à `"AATATGG"` et `"AAT";"ATGG"` correspond à `"AATGG"`.

À un niveau plus abstrait, on peut exprimer des vues concurrentes sur une même chaîne. Pour être une instance du modèle, la séquence doit être une instance de chaque vue du modèle. Par exemple, le modèle à deux vues `("TA",X)&(Y,"AT")` reconnaîtra les séquences qui commencent avec "TA" et finissent avec "AT", en incluant la reconnaissance du mot "TAT".
En pratique, la séquence d'un génome est obtenue par l'assemblage d'un grand nombre de segments séquencés chevauchants. Un modèle serait également à même de tracer de possibles erreurs d'assemblage dans un tel contexte.

Les répétitions en tandem ou les répétitions espacées sont des copies successives d'une même

[10] D={A,G,T}, H={A,C,T}, K={G,T}, S={C, G}, W={A,T}, Y={C,T}

entité qui sont très fréquentes dans les séquences génomiques. Logol introduit un constructeur de répétition gérant un compteur d'occurrences. Contrairement aux SRE, nous n'avons pas introduit ces compteurs comme un nouveau type de variable. Cela aurait introduit un niveau inutile de complexité qui n'est relayé par aucun besoin d'analyse de structures existantes connues en biologie. En Logol, une répétition est notée par un terme du type `(Entité Virgule Distance)+ NbOccur`, où le terme `Entité` désigne l'entité répétée, `Virgule` est le symbole de succession et `Distance` et `NbOccur` détaillent les contraintes sur les successions.
Par exemple, `("ACGA",[3,5] )+ [7]` s'appariera avec les chaînes contenant sept répétitions sans chevauchement du mot `"ACGA"`, apparaissant à une distance comprise entre trois et cinq, alors que `("ACGA";[3,5] )+ [7]` est un modèle similaire où les instances successives de `"ACGA"` peuvent se chevaucher du fait de l'utilisation du point-virgule.
Des propriétés telles que la prévalence de répétitions consécutives chevauchantes dans les répétitions en tandem n'ont jamais été testées dans des séquences biologiques du fait de l'absence d'une telle expressivité. Clairement, cela pourrait mener à une compréhension plus fine de la manière dont elles sont générées.

La combinaison de variables de chaîne, de successions chevauchantes et de vues permet d'exprimer n'importe quelle équation sur les chaînes et est potentiellement difficile à résoudre. Cependant, nous ne considérons que des séquences finies fixées et les modèles contiennent en pratique des contraintes sur la longueur des chaînes et un nombre limité de variables. Nous pensons qu'il existe dans ce contexte un large éventail de possibilités de développement de recherches sur la combinaison de techniques d'analyse syntaxique et de techniques de résolution de contraintes sur des chaînes [176].

Stratégies d'analyse et optimisation

Un point essentiel pour l'applicabilité de formalismes de récriture confrontés à des données réelles repose sur la partie souvent cachée du contrôle. L'analyse syntaxique d'un modèle complexe exige généralement un réglage fin de la stratégie d'analyse de façon à éviter à la fois des recherches inutiles (redondantes, ou localement insuffisamment contraintes) et la production de solutions non pertinentes. Ce point a été partiellement traité dans les Basic Gene Grammars [140] à l'aide d'une stratégie générale d'analyse tabulée (chart parsing) et traité de manière approfondie dans Genlang avec l'introduction d'un certain nombre de primitives de contrôle. Cependant, l'introduction de telles primitives déplace le statut de Genlang de celui d'un langage de modélisation vers un langage de programmation, et ceci est une limitation importante pour une large diffusion de l'outil. Un biologiste doit fournir un investissement initial important pour être autonome avec Genlang.

Notre approche est basée sur l'exploration et la mise en évidence des situations concrètes où les modèles biologiques ont besoin d'être raffinés au niveau de la stratégie d'analyse. Il s'agit ensuite de pouvoir les exprimer en termes de contraintes associées aux entités du modèle.
Dans la plupart des cas, les séquences ont évolué sous la pression de la sélection jusqu'à ce qu'elles aient atteint une configuration stable, localement optimale. Souvent, le biologiste a simplement besoin d'exprimer ce fait que les structures doivent être optimales en un certain sens. Par exemple,

les structures d'ARN doivent exhiber une forme de minimisation d'énergie dans leur repliement.

Pour illustrer les besoins sur un exemple simple, nous considérons la succession des gènes dans les séquences bactériennes. Un gène est défini comme une succession de codons, démarrant par un codon start (`"ATG"`, `"TTG"` ou `"GTG"`), et terminant par un codon stop (`"TAG"`, `"TAA"` ou `"TGA"`). Une règle de grammaire simple (`gene` $\longrightarrow$ `start, codons, stop`) reconnaîtra en fait toute suite de triplets de lettres entre un start et un stop, même si celle-ci se prolonge sur plusieurs gènes.

Pour éviter une telle combinatoire de reconnaissance, Logol[11] offre deux possibilités dans un tel cas.
On peut tout d'abord introduire une négation dans les contraintes sur l'entité "codons" (codons est une suite de triplets excluant les codons stop). Une seconde possibilité est d'introduire une contrainte d'optimisation dans le modèle (ici, pour un start donné, le stop doit avoir une position minimale).
Ensuite, si l'on veut obtenir la liste de tous les gènes d'une séquence, il faut que l'analyse puisse sauter d'un gène à l'autre sans considérer tous les chevauchements possibles. Par exemple, si l'on ne désire aucun chevauchement, on peut énumérer l'ensemble des solutions en ajoutant une contrainte sur la position du codon start, dont la distance à la fin d'une autre solution doit être minimisée.
De manière plus générale, il est possible en Logol d'ajouter une *contrainte d'optimisation* à toute entité sur une chaîne identifiée `SX`. Les contraintes d'optimisation portent sur les attributs des entités Logol (voir ci-dessous). Elles sont appliquées séquentiellement, en suivant leur ordre gauche-droite. Ainsi, choisir parmi les occurrences possibles celles qui ont la position de départ minimale par rapport à `SX`, la position de fin maximale ou la taille maximale, s'exprimera respectivement par `!@SX, !§SX` et `!#SX`.

Les bases de la syntaxe Logol

Logol permet de décrire des *modèles*, un modèle étant une représentation d'un langage, c'est-à-dire une représentation abstraite d'un ensemble de séquences. Nous donnons ici les principaux éléments du langage. La table 3.5 fournit quelques exemples pour les principales notions.

- Un modèle est composé d'*entités contraintes* (en syntaxe Logol : *Entités : Contraintes de chaînes : Contraintes de structure*).
- Une *entité* est une expression qui est soit une constante de chaîne, soit une variable de chaîne, ou un non-terminal, ou une répétition, ou un choix, ou récursivement de la forme *(Modèle)*.
- Les variables et les constantes de chaîne peuvent être précédées par un morphisme (p. ex. `+ comp("ATCC")` ou `- func(X)`). Un morphisme transforme une chaîne en analysant ces unités lexicales suivant la direction donnée et produit la concaténation des chaînes associées. Par exemple, si `comp` est défini à l'aide des règles

$$comp(\text{"}T\text{"}) \rightarrow \text{"}A\text{"},\ comp(\text{"}G\text{"}) \rightarrow \text{"}T\text{"},\ comp(\text{"}C\text{"}) \rightarrow \text{"}G\text{"},\ comp(\text{"}A\text{"}) \rightarrow \text{"}C\text{"},$$

[11] voir une expression Logol complète dans l'exemple `10` de la table 3.5

l'expression précédente `+comp("ATCC")` reconnaît la chaîne `"CAGG"` et l'expression `-comp("ATCC")` reconnaît `"GGAC"`.

- L'instance d'une entité dans une séquence est un couple <*chaîne, structure*>, où *chaîne* et *structure* sont des objets caractérisés par des attributs. L'objet *chaîne* pointe sur la partie de la séquence qui est une instance de l'entité, l'objet *structure* indique la manière dont l'analyse/appariement s'est effectué.
- Une *chaîne* est un objet qui représente une occurrence d'un mot dans une séquence. Chaque chaîne est caractérisée par cinq attributs : son nom, son contenu, sa position de départ, sa position d'arrivée et sa longueur. On définit un opérateurs d'accès pour chaque attribut.
- Une *structure* est un terme qui reflète l'appariement d'une chaîne sur la structure d'un modèle. Chaque structure est caractérisée par trois attributs : son nom, son contenu et son coût. L'accès à l'attribut coût utilise 4 opérateurs, suivant qu'on s'intéresse à une distance de Hamming ou une distance d'édition, et à des coûts absolus ou relatifs par rapport à la longueur de la séquence.
- Les modèles peuvent contenir plusieurs vues séparées par des $\&$.
  Une vue est une succession de séquences séparées par des `','` dans le cas non-chevauchant ou des `';'` dans le cas chevauchant.

Formellement, les grammaires Logol sont basées sur une extension des SVG que nous appelons Constrained String Variable Grammars (CSVG).

définition 9. Une *CSVG* est un octuple (Δ, Γ, Σ, N, V, S, F, P) où :

- Δ, Γ et Σ sont respectivement les alphabets des identificateurs de chaîne, des morphismes et des séquences ;
- $N = \bigcup N_k$ est l'ensemble des non-terminaux d'arité fixée k ;
- V est l'ensemble des variables de chaîne. L'ensemble $SV = \Sigma \cup V$ est l'alphabet étendu.
- $S \in N_0$ est l'axiome initial de la grammaire ;
- F est un ensemble fini de substitutions de Σ dans 2^Σ associées aux symboles de Γ. Les substitutions sont étendues de manière standard des lettres aux mots et signées : pour toute substitution f et tout mot aW de Σ^+, $f^+(aW) = f(a).f^+(W)$ et $f^-(aW) = f^-(W).f(a)$ (. est un produit d'ensembles) ;
- P est un ensemble de règles de production $A(X_1, .., X_k) \rightarrow \Phi$, avec $A \in N_k$, $X_i \in V$ pour $1 \leq i \leq k$ et Φ est un ensemble d'éléments de $R = (\mathcal{E} \times (\{,\} \cup \{;\} \cup \{\&\}) \times \mathcal{C})^*$, où $\mathcal{E} = (SV \cup (\{+,-\} \times \Gamma \times SV) \cup (\bigcup N_k \times V^k)$ et $\mathcal{C}$ est une contrainte.

Les règles de production permettent de dériver les mots du langage de manière habituelle, hormis le fait que la partie droite est un ensemble (de vues) dont tous les éléments doivent être dérivés. Les contraintes permettent d'établir la correspondance entre mots et chaînes. Une contrainte $\mathcal{C}$ peut être de deux types ; soit elle restreint l'ensemble des chaînes en bornant leur taille, leur position ou leur contenu sur un ensemble fixé de valeurs (contraintes de chaîne), soit elle restreint la façon dont s'apparie la chaîne au mot en bornant la distance entre eux (contraintes de structure). Enfin, il est possible d'accéder à la valeur de chaînes et d'utiliser le langage complémentaire d'un ensemble fini de chaînes.

1. *Modèle* `("CA"|"A"), "AATC", foo; foo → "TA"|"CT"` S'apparie par exemple avec `"AAATCTA"`. `foo` est un non-terminal. Un modèle de base est une succession d'entités contraintes. Ici c'est une succession contiguë (à cause de la ',') d'entités sans contraintes.
2. *Variable de chaîne* `X, "GAG", _, X,_` S'apparie par exemple avec la séquence `"GTTGAGAGGTTGA"` Ce modèle inclut les deux types de variables de chaînes (_, appelée variable muette, et `X`). Il reconnaît la séquence `X= "GTTGA"` et `X= "GTT"`. Comme en Prolog, les occurrences des variables muettes peuvent prendre des valeurs différentes. Elles sont utiles pour décrire des brêches (zones sans intérêt).
3. *Variable de chaîne avec morphisme* `X, -wc(X)` , où `wc("T")→"A", wc("G")→"C", wc("C")→"G", wc("A")→"T".` S'apparie avec le palindrome biologique `"TCGCGA"` pour `X="TCG"`. La seconde occurrence de `X`, est transformée par le morphisme inverse (sens −) `wc`.
4. *Succession chevauchante* `X, "GAG"; (_, X)` S'apparie par exemple avec `"GATTGAGATT"`, avec `X= "GATT"` et `_="A"`.
5. *Vues* `_,"ATCCGA",_ & _,"TAGTAT",_` S'apparie par exemple avec la séquence `"CC`**`TAGTATCCGA`**`TAC"`. Ce modèle établit que pour être admissible, une séquence doit simultanément contenir `"ATCCGA"` et `"TAGTAG"`, peu importe leurs localisations respectives.
6. *Contrainte de chaîne : longueur* `X : {#[2,4],_SX}, Y : #(#SX +1)` S'apparie par exemple avec `"AATCCAT"`, pour `X= "AAT"` et `Y= "CCAT"`. `_SX` dénotes l'identificateur se référant à l'occurrence de `X`. Ainsi `#SX` pointe sur la longueur de l'occurrence de `X`, et la contrainte `#(#SX +1)` requiert que la longueur de l'occurrence de `Y` soit plus longue d'un caractère par rapport à `X`. Ceci illustre le fait que l'on peut contraindre le domaine d'un attribut, et accéder à sa valeur pour poser des contraintes sur une autre entité.
7. *Contrainte de structure : distance de substitution* `X : :$[0,1], X : :$[0,1]` S'apparie par exemple avec la séquence `"AATT"` pour `X= "AT"` et `X= "TA"`. Description d'une répétition en tandem simple, où les copies d'un mot ancêtre inconnu ont pu diverger par au plus une mutation.
8. *Pseudonoeud* `_,X :#[5,8],_, Y :#[5,8],_, -wc X,_, -wc Y,_` Peut reconnaître le pseudonoeud de la figure 3.3.
9. *Contrainte de structure : distance d'édition* `X : #[2,4],_SX, X : :€[0,1], Y :#(#SX +1).` S'apparie avec `"ACGAGTTAT"`, pour `X="ACG"` et `Y="TTAT"`. Le modèle inclut une copie en tandem avec au plus 1 erreur. Il illustre le fait que `SX` se réfère seulement à la 1ère copie de `X` qui peut avoir une taille différant de celle de la seconde.
10. *Contraintes d'optimisation : position minimale* `(_, "ATG" :_SATG,@[3,25], ( :#3, ?~stop)+, stop,_) :!@SATG; stop→ "TAG"` Optimise la position des solutions. Sur `"ATGATGTATGGAATGTAGCATGCGGTAGG"`, il y a 2 solutions en [4,18] et [20,28] : [1,18], [4,28], [8,18], [13,28] sont incorrects.

TAB. 3.1 – Exemples de modèles Logol

STAN, Un premier analyseur pour un sous-ensemble de Logol

Nous avons développé avec Y. Mescam et A.-S. Valin une première implémentation d'un sous-ensemble restreint de Logol, STAN, qui a été transférée sur notre plate-forme de bioinformatique et publiée dans [161].
L'implémentation repose sur l'utilisation de la structure d'arbre des suffixes généralisé pour compiler les génomes à analyser et faciliter la recherche des occurrences de variables de chaîne. Elle utilise deux langages : Prolog et C. Prolog pilote la recherche tandis que le code C se charge du calcul de l'arbre des suffixes et de l'exécution des opérations de recherche sur celui-ci. Le programme Prolog prend en entrée une grammaire SVG qui décrit le motif à rechercher. Il en reconnaît les différents éléments (chaîne de caractère, variable de chaîne, disjonction, exclusion, espaces, etc...) et les traduit en une série d'opérations. Ces opérations sont transmises à l'application C, chaque opération correspond à une fonction de recherche dans l'arbre des suffixes : l'exécution d'appels de fonction sur l'arbre des suffixes est effectué selon un jeu d'instructions ordonnées. A l'échelle de l'arbre des suffixes, ces exécutions consistent à suivre un chemin constitué de noeuds internes : chaque appel de fonction est exécuté sur les noeuds solutions de l'appel de fonction précédent (le premier appel de fonction est exécuté sur le noeud racine de l'arbre des suffixes).

Afin d'accélérer les recherches, STAN a été parallélisé. La séquence à analyser est décomposée en plusieurs parties chevauchantes qui sont traduites en arbres des suffixes. La recherche s'effectue ensuite sur plusieurs arbres des suffixes (4 actuellement).

Du point de vue expressivité, STAN permet de décrire les éléments suivant en utilisant l'alphabet standard IUPAC décrivant des séquences d'ADN ou de protéines :

- Chaînes d'acides (résidus) (ACG), Choix de résidus ([ACG]) ou de chaînes ([AC|AG]) ;
- Exclusion de résidus (C) ;
- Gap de taille fixe ou variable : (x(2), x(5,8)) ;
- Motif avec erreurs de substitution (ATG :2) ou d'insertion/délétion (indel(ATCG, 0, 1, 1, 2)) ;
- Utilisation des variables de chaîne contraintes ($X : [10]$ ou $X : [120, 230]$) avec éventuellement des erreurs de substitution et l'inverse complémentation ($X, \tilde{X} : 1$)[12].

STAN permet également une recherche de motifs décomposés en 1 à 5 parties séparées par des espaces. Chaque partie est écrite en utilisant la syntaxe de motif de STAN, comme décrit précédemment, et les espaces sont utilisés afin de spécifier les distances minimum et maximum pouvant séparer deux parties consécutives du motif. Durant une recherche de motif décomposé, STAN tente de localiser chaque partie du motif dans une séquence d'ADN traduite dans ses six phases de lecture en tenant compte de la taille des espaces définis. Ce type de recherche permet d'obtenir un motif décomposé en plusieurs parties pouvant être localisées sur des phases de lectures différentes (mais sur le même brin d'ADN). Les motifs décomposés sont utiles en particulier pour localiser des gènes ayant une structure contenant des introns et des exons.
La figure 3.4 donne quelques comparaisons des performances de STAN par rapport à ses principaux concurrents, obtenues sur le génome d'*A. thaliana* pour des tailles croissantes de séquences extraites analysées. Dans le cas de motifs simples recherchés sans erreurs, des logiciels spécialisés comme Patmatch peuvent donner des résultats légèrement plus rapidement. Genlang a dans

[12] Actuellement, on ne calcule que pour une variable de taille 4 à 30.

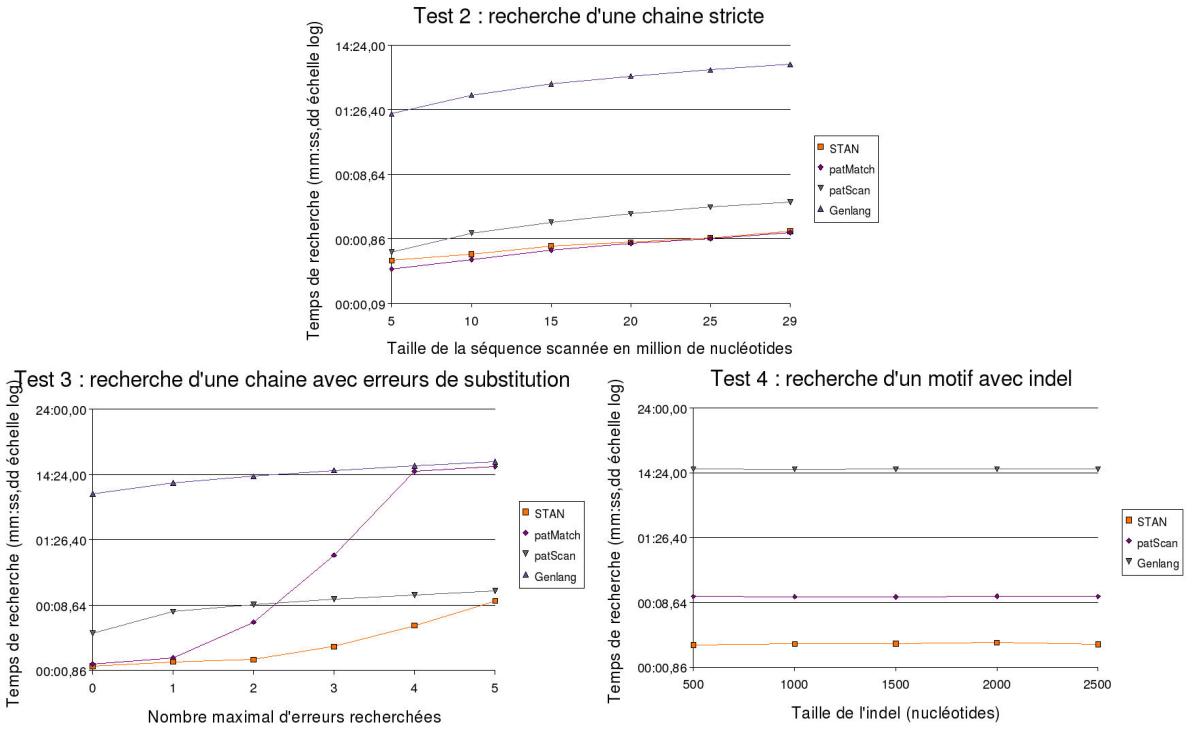


FIG. 3.4 – Performances comparées des analyseurs pour des tailles de séquences croissantes

tous les cas les temps de réponse les plus longs, mais possède l'expressivité la plus grande. Dès que l'on recherche des modèles avec erreurs ou des modèles comprenant des variables de chaîne, STAN a le meilleur comportement. Nous avons en particulier utilisé STAN dans la recherche d'un homologue rat d'un gène de récepteurs aux oestrogènes connu chez la souris mais non détectable par Blast [193], dans la recherche de gènes canins de récepteurs olfactifs à partir de motifs parsemés le long des gènes [180], et dans la recherche de transposons de type hélitron dans *Arabidopsis thaliana* [217].

Conclusion

Nous avons proposé un langage pour l'analyse de structures génomiques à partir principalement des travaux sur les SVG et en visant une expressivité améliorée. Les composantes clés du langage sont *les variables de chaîne avec erreurs, les analyses chevauchantes et concurrentes, et les primitives d'optimisation*. Ceci étend strictement les possibilités des SVG en préservant leur fondement sur des structures apparaissant naturellement dans les séquences biologiques. Bien sûr, Logol reste encore un chantier ouvert à compléter qui pourrait inclure d'autres caractéristiques pertinentes pour la modélisation de séquences, avec ce difficile challenge de rester pratique et opérationnel. Nous concluons par quelques directions de recherche que nous n'avons pu adresser dans notre travail et qui me semblent particulièrement importantes par rapport à cet objectif.

Leur propension à interagir entre elles est une caractéristique importante du monde des ma-

cromolécules biologiques. En particulier les molécules d'ADN et d'ARN peuvent s'hybrider entre elles de façon variée. Du point de vue de la modélisation, ceci signifie que nous devons considérer plusieurs séquences simultanément au lieu d'une seule.
Searls suggère [200] d'introduire un nouveau symbole dans les grammaires, le symbole de coupure, qui exprime une délimitation entre les diverses parties d'une chaîne en interaction. Ce symbole a pour effet de couper virtuellement la chaîne en ce point.
Une autre approche consiste plutôt à fournir à la fois un modèle avec des vues multiples et des séquences multiples, et exprimer les interactions à travers les variables de chaîne et les identificateurs de chaîne. Le point important qui reste à étudier est la sémantique à associer à de telles analyses multiples en relation avec les objectifs biologiques.
On peut ainsi être intéressé par un résumé du résultat de toutes les alternatives d'analyse possibles, par le filtrage des séquences répondant aux modèles ou par l'optimisation des appariements effectués sur l'ensemble des séquences. De ces choix dépendra la mise en oeuvre des analyses multiples.

Les vues et les variables de chaînes permettent d'établir des systèmes d'équations sur les patterns, au sens des langages de patterns (séquences de lettres et de variables). La puissance d'un résolveur de contraintes de chaîne dépend de la donnée d'une théorie équationnelle pertinente sur ces chaînes, juste suffisante pour prendre en compte les questions pratiques qui peuvent se poser sur les séquences génomiques. L'unification de patterns est un problème beaucoup plus difficile que l'unification standard sur les termes car il n'existe pas d'unificateur le plus général unique dans le cas général. Deux algorithmes existent pour résoudre les équations sur des patterns [114, 176] qui ont une complexité au moins DEXPTIME mais fournissent une base de travail saine pour des algorithmes plus spécialisés. L'analyse de modèle sur des séquences génomiques offre un contexte de travail idéal pour le développement de résolveurs de contraintes pratiques agissant en coordination avec les stratégies d'analyse standard.

Nous sommes en train de développer un analyseur de type DCG [168] pour la syntaxe complète de Logol, comme cela a été effectué avec GENLANG pour les règles SVG [63]. La mise au point d'analyseurs demeure une priorité et une nécessité forte pour la diffusion d'aspects avancés de la théorie des langages formels dans la communauté biologique.

Chapitre 4

Inférer des modèles syntaxiques sur des séquences génomiques

"Voulez-vous apprendre les sciences avec facilité ?
Commencez par apprendre votre langue...
Je regarde la grammaire comme la première partie de l'art de penser."
Etienne Bonnot de Condillac
Cours d'étude pour l'instruction du prince de Parme

L'inférence grammaticale est l'art de construire des grammaires à partir d'un échantillon de phrases. C'est un champ de recherche associé à la théorie des langages, qui a émergé à la fin des années 1960. Il s'agissait à l'origine de s'attaquer aux problèmes d'apprentissage en reconnaissance des formes. La communauté associée a toujours été curieusement relativement restreinte si on la compare à celle qui s'est intéressée à la théorie des langages et ses applications. Notre propre pratique des problèmes rencontrés dans le cadre de nos travaux en génomique nous pousse cependant à défendre ce champ comme une opportunité claire d'enrichir et de relancer les travaux de recherche en théorie des langages. Notre conviction repose sur le fait qu'en inférence grammaticale, comme dans toute forme d'apprentissage automatique explicite, la représentation retrouve un rôle clé. Dans le souci utile de rationaliser les concepts nécessaires, la théorie des langages a recherché systématiquement des classes d'équivalence entre représentations de classes de langage. Ce faisant, elle a du même coup appauvri la variété des tentatives dans ce domaine. En inférence grammaticale, la représentation est importante pour au moins trois raisons :

- Le but est d'obtenir un modèle, voire une explication de la manière dont les séquences ont été générées. Chaque représentation offre un point de vue différent et donc un modèle différent, même si le langage généré est identique.
- L'optimisation d'un modèle repose toujours sur une notion de complexité proche de la complexité de Kolmogorov : on cherche à minimiser la taille de la machine ayant généré le langage. Cependant, cette complexité est intimement liée aux descriptions admissibles pour ces machines et elle n'a donc de sens que par rapport à la classe de représentations visée.
- Enfin, d'un point de vue algorithmique, l'apprentissage est un problème de recherche combinatoire dans un espace d'hypothèses. Certaines représentations sont beaucoup plus inté-

ressantes que d'autres quant à la structure et à la définition d'opérateurs de recherche dans cet espace d'hypothèses.

Nos propres travaux nous ont conduits à la fois à étudier différentes représentations de langages, et à introduire des contraintes propres au domaine de la biologie moléculaire.

4.1 La problématique de l'inférence grammaticale

On appelle inférence grammaticale l'apprentissage automatique d'un modèle de langage à partir d'un échantillon fini des phrases du langage qu'il accepte (instances positives) et éventuellement d'un échantillon fini de phrases n'appartenant pas à ce langage (instances négatives). Spécifier complètement un problème d'inférence grammaticale suppose de :

- définir la classe des langages acceptés ;
- définir une représentation pour cette classe (grammaires formelles, automates, expressions) ;
- définir une relation d'ordre (relation de généralité) sur ces représentations, compatible avec l'inclusion sur les langages ;
- définir les conditions de présentation des phrases d'apprentissage ("oracle" répondant aux questions de l'algorithme, présentation en bloc des instances ou incrémentale) ;
- définir un critère d'acceptation des solutions en fonction des instances, basé entre autres sur l'acceptation des instances positives et le rejet des instances négatives dans les langages associés aux solutions.

En pratique, l'apprentissage de grammaires est toujours traité comme un problème d'optimisation. Il s'agit de minimiser un score qui peut tenir compte de la taille de la grammaire inférée (complexité de la représentation) et du degré de couverture des instances d'apprentissage (on peut tenir compte d'un certain niveau de bruit). L'inférence grammaticale a été appliquée dans un certain nombre de cas aux séquences d'ARN ou de protéines. On trouvera une bonne revue dans [191].

4.2 Inférence de grammaires algébriques

J'ai commencé par m'intéresser à la classe des langages algébriques, qui a potentiellement un très grand champ d'applicabilité grâce à sa capacité de décrire des structures auto-imbriquées. Ceci a donné lieu à la soutenance de la thèse de J.Y Giordano [93] et a fait l'objet d'un transfert via un postdoctorat industriel avec la société Genset.

Il y a eu très peu de travaux s'attaquant à la classe entière des langages algébriques. Citons essentiellement ceux de Tanatsugu [212], qui propose un algorithme non polynomial construisant des grammaires linéaires sur les structures auto-imbriquées observées qui sont ensuite assemblées pour produire la grammaire algébrique finale.

Nous nous sommes situés pour notre part dans le cadre générique de l'espace des versions [152], où l'on peut réduire l'apprentissage de grammaires algébriques à un problème d'*énumération dans un espace d'ensembles de règles partiellement ordonné par une relation de généralisation compatible avec l'inclusion sur les langages*.

L'algorithme d'apprentissage à partir d'exemples et de contre-exemples est très simple dans un tel cadre et nous le décrivons rapidement.
On maintient l'ensemble S des solutions les plus spécifiques (langages les plus petits) et l'ensemble G des solutions les plus générales (langages les plus grands) acceptant les exemples et ne contenant pas les contre-exemples, en maintenant l'invariant suivant : tout langage de S (respectivement G) est inclus (resp. contient) dans au moins un langage de G (respectivement S). La donnée d'un exemple de mot w du langage cible permet de généraliser des éléments de S (pour accepter w) et supprimer des éléments de G (ceux n'acceptant pas w) alors qu'un contre-exemple permet de spécialiser les éléments de G et supprimer des éléments de S. Une telle représentation présente un certain nombre d'avantages, dont celui de représenter de manière compacte l'ensemble des solutions (et non pas une seule solution comme c'est généralement le cas). Elle offre également un critère de convergence associé simple ($S = G$ est un singleton).

La principale difficulté est le choix de la relation de généralité ordonnant les grammaires de l'espace des versions. Un choix naturel est **l'inclusion faible**, qui correspond à l'inclusion des langages reconnus par les grammaires. Cette relation est inaccessible (indécidable) dans le cas des grammaires algébriques [7].
La plupart des auteurs proposent de travailler en conséquence sur des données plus riches, où les arbres de dérivation des mots en exemple sont totalement [78, 190] ou partiellement connus [219]. Fondamentalement, il s'agit alors d'exploiter le fait que l'ensemble des arbres de dérivation d'une grammaire algébrique forme un ensemble régulier.
Notre approche a plutôt été d'utiliser sur l'espace des grammaires une relation plus faible, basée sur ces ensembles d'arbres de dérivation, réduisant de ce fait des grammaires équivalentes du point de vue des structures de dérivation engendrées à un seul représentant.
Une grammaire G_1 est **structurellement incluse** dans une autre G_2 si les mots engendrés par G_1 le sont aussi par G_2 et si l'ensemble des arbres de dérivation qui peuvent être obtenus à partir de G_1 est contenu dans l'ensemble des arbres de dérivation de G_2.
La relation d'inclusion structurelle, contrairement à la relation d'inclusion faible, est calculable en temps exponentiel sur la classe des grammaires algébriques, ou, ce qui revient au même, l'inclusion faible est calculable sur les grammaires algébriques parenthésées. On peut en fait aller plus loin en normalisant la forme des grammaires. Nous avons choisi la forme normale de Chomsky de grammaires inversibles dans notre cas, c'est-à-dire où toutes les règles sont de la forme
Non-terminal $\leftarrow$ (Non-terminal Non-terminal). ou Non-terminal $\leftarrow terminal.$ et où pour toute partie droite de règle, il existe une seule partie gauche.
On sait que l'ensemble des grammaires inversibles a le même pouvoir d'expression que l'ensemble des grammaires algébriques parenthésées [102] (on connaît une transformation exponentielle depuis une grammaire algébrique parenthésée vers une grammaire inversible). On décrit donc bien le même espace en se restreignant à ces grammaires avec l'avantage de disposer d'un algorithme polynomial dans ce cas pour tester l'inclusion structurelle entre G_1 et G_2 : il suffit d'établir une substitution des non-terminaux N_1 de G_1 vers l'ensemble des parties 2^{N_2} des non-terminaux de G_2 telle que toute substitution d'une règle de G_1 apparaisse dans G_2.

Une fois l'espace d'hypothèses ordonné, on cherche à le parcourir efficacement au moyen d'opérateurs. Ces opérateurs de déplacements élémentaires vont permettre de converger progres-

sivement vers une solution, en produisant une suite ordonnée de grammaires jusqu'à ce qu'elles respectent les contraintes imposées par les instances d'apprentissage. On se heurte alors à une autre difficulté de taille, le choix d'opérateurs assurant la complétude du parcours et évitant de produire des grammaires équivalentes.
Nous n'avons pas pu mettre au point un ensemble complet d'opérateurs, problème qui s'est révélé ardu, même dans le cadre restreint des langages réguliers. Nous avons par contre proposé deux opérateurs en supposant fixé le nombre de non-terminaux de la grammaire. Cette restriction est intéressante car le nombre de non-terminaux est un indice pertinent de la complexité de la grammaire. Il est possible de chercher à minimiser ce nombre en explorant itérativement l'ensemble des grammaires pour un nombre fixé de non-terminaux croissant.
Les deux opérateurs pour explorer l'espace de recherche, tout en maintenant le nombre de non-terminaux, sont un opérateur de *substitution sur la partie gauche des règles* et un opérateur d'*ajout/suppression de règles*. Pour la substitution des symboles non-terminaux, nous avons utilisé l'inclusion des contextes associés, où la notion de contexte se réfère aux dérivations admissibles contenant le symbole non-terminal :

$$Contexte(N) = \{(u, v) \in (V \cup \Sigma)^* \times (V \cup \Sigma)^* | S \stackrel{*}{\Longrightarrow} uNv\}.$$

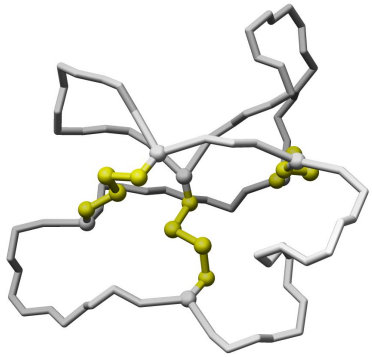
La thèse de J.Y. Giordano [93] propose un algorithme pour tester l'inclusion de contextes en O(pn^4), où p est le nombre de règles et n le nombre de non-terminaux.

Bien que réduisant fortement la complexité de l'énumération, ce système ne nous a cependant pas permis d'aboutir à des résultats d'identification, ni au traitement de tailles de grammaires suffisantes en dehors de cas d'école (3 à 5 non-terminaux).

Nous avons également essayé une relation d'inclusion encore plus faible sur les grammaires, la **couverture de Reynolds**. Une grammaire G_2 est une couverture de Reynolds d'une autre G_1 s'il existe une fonction de renommage des non-terminaux de G_1 vers ceux de G_2 telle que la transformée des règles de production de G_1 par ce renommage soit incluse dans G_2. Nous nous sommes posés dans ce contexte le problème d'explorer l'ensemble des grammaires dont la grammaire universelle est une couverture de Reynolds. Cet espace est structuré en demi-treillis reliant des couches de grammaires à nombre fixé de non-terminaux. Partant initialement de la grammaire universelle, on applique itérativement une opération de spécialisation permettant de passer d'une couche à l'autre.
Cette approche s'applique aussi bien aux grammaires régulières qu'algébriques. L'opération de spécialisation prend une grammaire à n non-terminaux et génère toutes les grammaires à $n + 1$ non-terminaux issues de la fission du dernier non-terminal introduit. On commence par choisir une partie des règles de production de la grammaire à spécialiser contenant le dernier non-terminal choisi X_n et on génère pour chacune de ces règles un certain nombre de copies différentes en substituant au moins un X_n par un nouveau non-terminal X_{n+1}. Si on parcourt l'ensemble des parties possibles et l'ensemble des substitutions possibles, on obtient une stratégie complète d'exploration de l'espace d'hypothèses qui évite d'arriver à une même grammaire par un très grand nombre de chemins. Hélas, si la redondance est réduite, il n'en existe pas moins un nombre extrêmement élevé de solutions concurrentes possibles à chaque niveau. Ceci impose en pratique le recours à une recherche heuristique en faisceau qui semble très délicate à mettre au point.

Du fait de l'ampleur et la difficulté du problème, je suis revenu dans la suite de mes recherches en inférence grammaticale au cas plus simple où on s'intéresse juste au *problème de la généralisation* et où la classe acceptée est la *classe des langages rationnels représentés par automates finis*. Cependant, les questions soulevées par l'étude de l'inclusion structurelle demeurent d'actualité et forme une base riche et solide pour de futures recherches en inférence de grammaires algébriques. D'une manière plus générale et au delà de l'apprentissage, c'est la théorie des langages elle-même qui peut bénéficier de ces études, car elles permettent de préciser la structure de l'espace des grammaires et le noyau syntaxique nécessaires à la génération des langages algébriques.

J'ai néanmoins abordé un problème nécessitant une expressivité algébrique quelques années plus tard, dans le cadre d'une action nationale (ACI Genogrid, collaboration avec C. Gourjeon de l'IBCP à Lyon), sur un problème difficile de prédiction. J'ai encadré la thèse d'I. Jacquemin sur la prédiction de certaines liaisons (ponts disulfures) entre des sites distants sur une séquence de protéine [113].



Le repliement des protéines repose sur des interactions faibles ou covalentes entre les composants de base des protéines (les acides aminés), dont une des formes les plus courantes est la formation des ponts disulfures entre les groupements thiol de la cystéine, un acide aminé particulier. Ce type de liaisons est particulièrement structurant. On montre sur l'image ci-contre l'exemple d'un polypeptide cyclique, le cyclotide, qu'on retrouve dans certaines plantes comme la violette. Les 3 ponts disulfures que contient ce polypeptide (en jaune sur l'image) de 30 acides aminés lui confère une stabilité chimique et biologique exceptionnelle, qu'on cherche à exploiter pour la mise au point de médicaments.

Les ponts disulfures, liaisons croisées entre des chaînes de protéines différentes ou entre des régions d'une même chaîne, sont formés par l'oxydation de cystéines. Du point de vue des problèmes de prédiction associés, on peut distinguer deux tâches de difficulté croissante :

- Le problème de *localisation des cystéines oxydées* impliquées dans un pont. D'un point de vue plus formel, il s'agit de classer dans un mot toutes les lettres C qu'il contient en deux classes, celles qui font partie d'une liaison et les autres. Ce problème est assez bien maîtrisé. Mucchielli-Giorgi et al.[156] annoncent par exemple des taux de prédiction correcte de 84%.
- Le problème de *prédiction de la structure des ponts disulfures* de la protéine. D'un point de vue plus formel, il s'agit cette fois de prédire les couples de C appariés dans le mot. A l'inverse du problème précédent, il n'existe actuellement pas d'algorithme satisfaisant capable de prédire les ponts disulfures dans la protéine.

Nous avons exploré deux pistes de recherche dans ce cadre. La première idée [112], tirée d'une approche qui a été proposée sur un plan théorique par Takada, a été d'explorer une méthode d'**inférence grammaticale sur un langage de contrôle**. On travaille pour cela non pas sur les séquences initiales, mais sur les chaînes de production d'une grammaire universelle de plus haut niveau (langages de Szilard). Plus précisément, nous partons d'une grammaire "universelle", ca-

pable de reconnaître toutes les liaisons possibles plus d'autres, non pertinentes du point de vue de l'application. Chaque règle de cette grammaire est numérotée et on transforme les séquences originelles par le code des règles nécessaires pour l'analyse de ces séquences avec la grammaire. L'inférence est alors conduite sur ces séquences recodées. Le grand intérêt de cet artifice est qu'il permet théoriquement de grimper artificiellement dans la hiérarchie de Chomsky. On peut attaquer une classe de langages plus large que celle du régulier tout en conservant une méthode d'inférence régulière.

A priori, modéliser un pont cystéine peut s'effectuer à l'aide d'une grammaire algébrique, en utilisant une règle du type $S \leftarrow' C'S'C'S$, qui met en relation les deux cystéines. Celle-ci vient compléter une règle de reconnaissance de cystéine standard, $S \leftarrow `C''S$, qui analyse chaque cystéine indépendemment. Notons qu'une telle grammaire est non déterministe et même ambiguë, mais cela n'est pas gênant dans le contexte d'apprentissage d'un langage de contrôle sous forme d'un automate déterministe cette fois, qui sélectionnera à chaque fois la règle à appliquer. Hélas, les grammaires algébriques ne peuvent a priori décrire que des dépendances imbriquées ("bien parenthésées") et ce n'est pas toujours le cas dans les structures de protéines.
Prenons l'exemple de la super-famille des knottines, à laquelle appartient le cyclotide précédent. Les ponts disulfures que les protéines de cette familles exhibent ont la structure schématisée sur la figure 4.1. D'un point de vue grammatical, les dépendances observées sont croisées et nécessitent donc théoriquement des grammaires dépendantes du contexte.

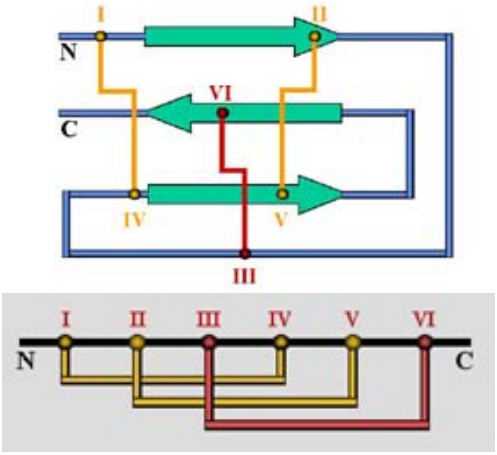


FIG. 4.1 – Structure chevauchante des ponts disulfures associés aux protéines de la super-famille des knottines

Nous avons contourné la difficulté en utilisant des règles spécifiques de certains motifs comportant plusieurs ponts cystéines. Dans notre exemple, on pourrait décrire cette structure par une règle $S \leftarrow' C'T'C'T'C'T'C'T'C'T'C'S$. Rien n'indique dans la règle les appariements effectués. C'est le numéro de la règle qui fournira la sémantique implicite associée d'appariement 1-4, 2-5 et 3-6. Si d'autres structures d'appariements à 6 cystéines existent, il faudra dupliquer la règle, en lui attribuant un nouveau numéro. Les productions de la grammaire étant contrôlées, utiliser l'une ou l'autre règle n'aura pas du tout la même signification.

La thèse a montré de nombreuses difficultés dans la mise en place pratique de ce schéma, notamment dans la génération de contre-exemples et le traitement du bruit. La génération de contre-exemples pose en général un problème en biologie où il est difficile de spécifier la négation d'un concept (qu'est-ce qu'un "non pont" cystéine ?). Dans notre cas, nous faisons l'hypothèse réaliste d'un phénomène de compétition réel, tel que les cystéines impliquées dans un pont donné ne peuvent l'être avec un autre pont non observé. Les contre-exemples ont donc été construits à l'aide d'échanges de 2 ponts, avec suppression éventuelle.
Mais nous n'avons pu aboutir à des résultats de prédiction satisfaisants par inférence grammaticale dans le temps de la thèse (on pourra cependant consulter [111]).

Il reste cependant énormément de voies de recherche à explorer dans le domaine de l'inférence de grammaires algébriques. J'aimerais terminer cette section en évoquant une piste qui me semble particulièrement prometteuse, sur un problème plus spécifique pour lequel existent des avancées récentes de bonne augure. Il s'agit du problème du plus petit code basé sur une grammaire [127], qui consiste à représenter une séquence à l'aide d'une grammaire algébrique la générant. Contrairement au problème de l'inférence, une seule séquence est disponible et le but est simplement de comprimer celle-ci en repérant les répétitions et pas de fournir une quelconque explication structurelle. Sur l'ADN, c'est la compression des génomes qui est visée. Ces travaux rejoignent aussi la vision développée dans cette synthèse d'un nécessaire appui de la modélisation syntaxique sur un fort composant lexical, qui puise son sens dans les répétitions naturelles produites dans les séquences biologiques.
Fondamentalement, on optimise un critère de type complexité de Kolmogorov, en se restreignant à des grammaires algébriques. En travaillant sur une séquence qui est la concaténation des instances positives d'apprentissage, séparées par un caractère spécial, et si l'on choisit la somme des tailles des membres droits des règles de la grammaire comme critère à optimiser, les méthodes de résolution du problème du code basé sur une grammaire sont des méthodes applicables en inférence grammaticale.
Comme dans les travaux que nous venons de présenter, il existe deux voies possibles de résolution. Dans la première, on se donne une grammaire universelle puis on compresse l'arbre d'analyse de la séquence par cette grammaire. Ceci a été utilisé en particulier dans la compression d'images [124] puis l'encodage de sources quelconques [35]. La deuxième voie, appelée simplement *problème de la plus petite grammaire* [36, 37], nous semble la plus riche du point de vue des idées et algorithmes qui ont été développés. Dans ce cas, on cherche la plus petite grammaire algébrique représentant le langage reconnaissant exactement la séquence. La taille d'une grammaire est la somme des symboles des parties droites de ses règles. On code ensuite la grammaire avec un algorithme de compression de la chaîne de ses parties droites.
Du fait de l'objectif de compression, les auteurs recherchent en général un algorithme rapide (linéaire en fonction de la taille de la séquence à coder), voire idéalement "online", c'est-à-dire effectuant une seule lecture gauche droite de la séquence entière. Les séquences sont supposées d'entropie élevée. En ce qui concerne les données génomiques, l'enjeu est celui de l'explicitation, sur une séquence composite, des différentes zones et de leur structure. Le temps de codage importe peu, l'entropie est très variable et un bon taux de compression sur une partie seulement de la séquence est déjà intéressant. Moyennant ces spécificités qui ne sont pas toujours prises en compte pour traiter les séquences génomiques, le domaine de la compression basée sur des grammaires

offre des algorithmes performants qui explorent des classes de grammaire variées constituant une base d'idées originales pour l'inférence grammaticale.
Pour garantir de générer une seule séquence et ne pas explorer de grammaire a priori non pertinente, on impose généralement deux propriétés à la représentation grammaticale : les grammaires doivent être déterministes (pour tout non-terminal, il existe exactement une règle avec ce non-terminal en partie gauche) et ne doivent pas contenir de symbole inutile (tout non-terminal apparaît dans la dérivation d'au moins un mot du langage et il n'y a pas de dérivation dans le vide). Ceci implique en particulier qu'il n'y ait pas de cycle dans la grammaire. Chaque méthode a ensuite des conséquences sur la forme des grammaires explorées, et il est intéressant de les étudier selon ce point de vue qui peut apporter des propriétés pertinentes pour la problématique d'inférence grammaticale.
Ainsi, si l'on considère la méthode LZ78 de compression de Ziv et Lempel [226], on observe que par analyse gauche droite du plus petit mot n'apparaissant qu'après une position donnée, elle génère essentiellement une grammaire régulière, mise à part la règle associée à l'axiome qui est une suite de non-terminaux. La méthode MPM employée par Kieffer et al. [126] se base quant à elle sur une segmentation récursive de la séquence qui génère l'arbre de dérivation de la séquence de manière descendante. Si on considère uniquement des partitions en 2 à chaque étape, on obtient ainsi une restriction à des grammaires sous forme normale de Chomsky. La méthode Sequitur [158] et ses variantes se restreignent à une classe plus complexe de grammaires, dites "irréductibles" [224] : il y a une seule règle ayant un non-terminal donné en partie gauche ; il n'y a pas de dérivation dans le vide ni de symboles inutiles ; il n'y a pas deux non-terminaux équivalents en terme de leur expansion ; chaque symbole non-terminal différent de l'axiome apparaît au moins deux fois en partie droite des règles ; et enfin, les seules répétitions de paires de symboles dans la grammaire doivent être chevauchantes. On retrouve cette classe de grammaires dans bon nombre de méthodes (p. ex. [136, 14, 127]).

4.3 Inférence d'automates d'états finis

Le problème qui a fait l'objet des travaux les plus nombreux en inférence grammaticale est très certainement celui de l'inférence de langages réguliers. Se sont des langages aux propriétés sympathiques pour lesquels il a existé très tôt des algorithmes d'inférence efficaces [11, 166]. Nous avons vu dans le chapitre précédent que cette classe de langages permettait déjà de décrire un certain nombre de phénomènes sur les séquences biologiques, en particulier sur les protéines. Il était donc naturel pour nous d'essayer de progresser sur ce problème.

Au moment où nous avons démarré nos travaux, la compréhension théorique fine de l'espace de recherche associé à l'inférence régulière commençait à se dévoiler. Il existe une relation d'ordre de généralité naturelle sur les automates, induite par la fusion d'états. Fusionner deux états consiste à les rendre équivalents, en les remplaçant par un état unique qui possède l'union des transitions entrantes et sortantes de cet état. Toute fusion d'états dans un automate mène à un automate (appelé automate dérivé) reconnaissant un langage plus général ou équivalent au langage reconnu initialement.
Si de plus on prend comme critère d'acceptation la complétude structurelle (i.e. toute transition

d'un automate solution est exercée et tout état final est utilisé dans l'analyse de l'échantillon d'apprentissage), on montre que l'espace de recherche de toutes les solutions est un treillis [67]. Celui-ci peut être construit à partir d'un élément nul, l'automate canonique reconnaissant uniquement les instances positives.
Les éléments du treillis sont dérivés de l'élément nul par une fonction correspondant à la fusion de ses états. L'élément universel du treillis est l'automate universel, reconnaissant n'importe quelle suite de caractères. On peut restreindre encore l'espace de recherche si l'on s'intéresse uniquement aux automates déterministes. Dans ce cas, on remplace l'automate canonique initial par l'arbre accepteur des préfixes (PTA) comme élément nul du treillis. Celui-ci peut être dérivé de l'automate canonique en fusionnant les états partageant les mêmes préfixes. La figure 4.3 illustre les principales notions utilisées en inférence régulière. L'opération de base qu'est la fusion peut être prolongée en déclenchant des fusions supplémentaires tant que l'automate n'est pas déterministe. On parle alors de *fusion déterministe*.

J'ai proposé avec F. Coste une approche de l'inférence régulière par gestion d'un ensemble de contraintes dynamiques sur les équivalences d'états [50, 53, 51, 48, 52]. Explorer l'espace des automates du treillis est équivalent à explorer l'espace des fusions d'états de l'élément nul du treillis, ou encore l'espace des partitions sur ces états puisque chaque fusion peut être caractérisée par l'ensemble des états qu'elle fusionne.
Nous avons particulièrement travaillé sur la prise en compte d'un échantillon d'instances négatives, c'est-à-dire de mots n'appartenant pas au langage cible.

Nous avons besoin d'introduire quelques définitions pour préciser nos principaux résultats.
On note q_π le bloc de la partition π contenant l'état q. On note I_+ l'ensemble des séquences données comme exemples (échantillon positif) et I_- l'ensemble des séquences données comme contre-exemples (échantillon négatif). L'automate $APTA(I_+, I_-)$ est simplement un automate étendu pour prendre en compte les deux échantillons, avec deux ensembles d'états finals F^+ et F^- pour la reconnaissance des instances positives et négatives.
On définit une relation binaire $\rightarrow$ sur des paires d'états, appelée *fusion déterministe*. Formellement, on a

$$(q_1, q_2) \rightarrow (q_1', q_2') \text{ si et seulement si } \exists a \in \Sigma \;\; \delta(q_1, a) = q_1' \wedge \delta(q_2, a) = q_2'. \tag{4.1}$$

En s'intéressant plus particulièrement à l'espace des fusions, on peut introduire les ensembles suivants, qui caractérisent les solutions du problème d'inférence :

- f_{Ok} (resp. f_{Nok}) : ensemble des fusions consistantes (resp. non consistantes) avec l'échantillon négatif ;
- F_{Ok} (resp. F_{Nok}) : ensemble frontière de f_{Ok} (resp. f_{Nok}), c'est à dire des fusions les plus grandes (resp les plus petites) de celui-ci.

Nous avons montré que tout automate déterministe compatible avec les échantillons I_+ et I_- était l'automate dérivé de $PTA(I_+)$ en appliquant les fusions d'états qui respectent les contraintes suivantes :

$$q_\pi^+ \neq q_\pi^- \qquad \forall q^+ \in F^+, \forall q^- \in F^- \tag{4.2}$$

$$q'_{1\pi} \neq q'_{2\pi} \Rightarrow q_{1\pi} \neq q_{2\pi} \quad \forall q_1', q_2' \in APTA(I_+, I_-) \text{ tels que } (q_1, q_2) \rightarrow (q_1', q_2') \tag{4.3}$$

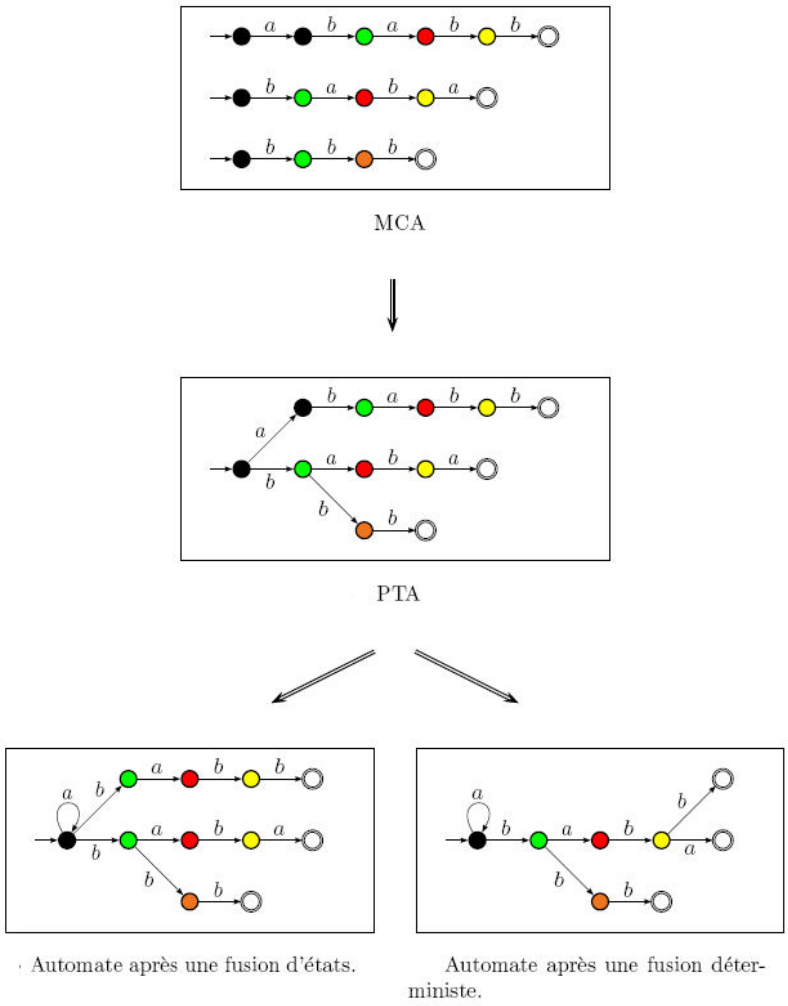


FIG. 4.2 – Automate canonique maximal, arbre accepteur des préfixes, fusion et fusion déterministe d'états

On notera que les contraintes du type 4.3 peuvent être inversées sous forme de contraintes d'égalité $q_{1\pi} = q_{2\pi} \Rightarrow q'_{1\pi} = q'_{2\pi}$. De plus, le système de contraintes peut être simplifié pour supprimer les transitivités (p. ex. $\{1 \neq 2,\ 1 \neq 2 \Rightarrow 3 \neq 4\}$ est réduit à $\{1 \neq 2,\ 3 \neq 4\}$). Nous avons démontré que les contraintes de type 4.2 (après simplifications) caractérisent F_{Nok} et peuvent être représentées par un graphe sur l'ensemble des états (les arcs joignant les états dont la fusion provoque l'obtention d'un automate non consistant avec I^-) :

proposition 5. *L'ensemble* $F_{Nok}(PTA(I^+, I^-))$ *des fusions non consistantes d'états est*

$$\{\ (q_1, q_2) \textit{ tels que } (q_1 \in F^+ \wedge q_2 \in F^-) \textit{ ou } \ \exists f \in F_{Nok}(PTA(I^+, I^-))\ \ (q_1, q_2) \stackrel{*}{\rightarrow} f)\}.$$

Si p est le nombre d'instances positives, n le nombre d'instances négatives et m la longueur moyenne des instances, l'algorithme est dans le pire des cas en $O(pnm)$.

La relation entre $F_{Nok}(PTA(I^+))$ et l'ensemble des automates solutions peut alors être formulée en terme de coloriage. Nous rappelons la définition d'un coloriage.

définition 10. Un graphe $G = (S, A)$ est k-coloriable si et seulement si il existe une application γ de S dans l'ensemble $\{1, 2, \ldots, k\}$ telle que $(u, v) \in A$ entraîne $\gamma(u) \neq \gamma(v)$ (une telle fonction est appelée k-coloration)

Les automates solution vérifiant le critère de consistance sont tels qu'il existe une k-coloration sur $F_{Nok}(PTA(I^+))$ telle que tous les états de chaque bloc de la partition des états sont de la même couleur.

Si l'on désire satisfaire en plus le critère de minimalité sur les automates, la k-coloration doit être minimale (et le nombre chromatique du graphe correspond au nombre d'états des automates solution). Bien entendu, le problème du coloriage demeure NP-complet dans sa généralité, mais on a réduit le problème à un cadre qui permet de profiter des progrès sur un problème générique. L'algorithme 4 donne le schéma de l'approche que nous avons proposé.

Algorithm 4 Algorithme de fusion d'états étendu pour le traitement d'incompatibilités.

ESMA(A)
($s1$, $s2$) ← ChoisirÉtatsAFusionner(A) ;
if Il existe un couple d'états à fusionner ($s1$, $s2$) **then**
  if ($A' \leftarrow A[s1 = s2]$) /* On fusionne les états et on en propage les conséquences */ **then**
    ESMA(A')
  end if
  /* Extension permise par la gestion de F_{Nok} */
  if ($A' \leftarrow A\ [s1 \neq s2]$) /* On ajoute une contrainte et on en propage les conséquences */ **then**
    ESMA(A')
  end if
else
  Output A, si A est solution
end if
end

En collaboration avec C. Rétoré, nous avons enfin abordé le cas des grammaires lexicalisées, plus particulièrement adaptées aux traitements de la langue naturelle. Ce dernier travail a fait l'objet d'un rapport Inria [160], où j'ai essayé de faire le point sur les liens entre logique et inférence grammaticale. Deux approches y sont présentées, l'une où l'on cherche à caractériser le type des mots du langage, l'autre où on cherche à caractériser l'automate du langage. On peut en effet soit chercher à généraliser au niveau des éléments des séquences (quels types de voisins admet un mot donné ?), soit généraliser au niveau de l'analyseur des séquences (quels suffixes sont possibles après un ensemble de préfixes donné ?). Dans tous les cas, la recherche de solutions admissibles dans une classe donnée de langages est réduite au problème de l'unification d'un ensemble de termes. Ce point de vue est développé dans le contexte des grammaires lexicales (grammaires catégorielles) en utilisant une unification classique sur des termes typés et dans le cas des grammaires régulières (automates finis) en utilisant une unification rationnelle.

4.4 Le passage de la théorie à la pratique : inférence de modèles syntaxiques sur des séquences de protéines

Tous ces travaux ont été ensuite approfondi pour faire face aux contraintes issues de nos applications en bioinformatique. Nous avons exploré plus avant les problèmes d'inférence grammaticale lorsque les séquences sont des protéines. Outre la thèse d'Ingrid Jacquemin que nous avons déjà évoquée dans la section sur l'inférence de grammaires algébriques (et qui portait sur la reconnaissance de ponts cystéines), deux thèses ont été soutenues sur ce volet.

J'ai d'abord encadré avec F. Coste la thèse de D. Fredouille [83], thèse qui s'est attaquée au problème délicat de l'**inférence d'automates non déterministes**. Les automates non déterministes (NFA) sont en général délaissés par la communauté qui travaille sur les langages formels, car on sait transformer facilement un NFA en automate déterministe, qui sera beaucoup plus facile à manipuler[1]. Cependant, le nombre d'états d'un NFA nécessaire pour représenter un langage régulier peut être exponentiellement plus petit que celui de l'automate déterministe minimal pour ce langage. Cet avantage révèle en fait une autre caractéristique très intéressante des automates non déterministes : ils permettent d'analyser de façon beaucoup plus souple une structure régulière et possède donc un pouvoir explicatif plus fort. La figure 4.3 illustre cet aspect en donnant trois versions d'automates pour un même langage $\Sigma^*0\Sigma^2$, un automate non déterministe, un automate déterministe minimal, et en bas un automate non déterministe à états résiduels.

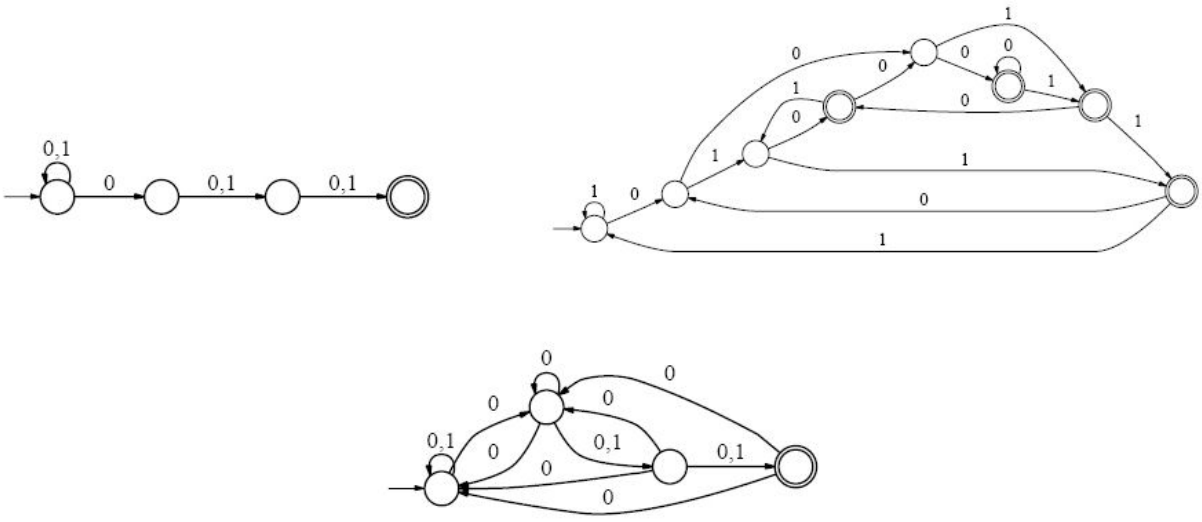


FIG. 4.3 – Trois représentations par automate fini d'un même langage $\Sigma^*0\Sigma^2$

S'il s'agit de construire des modèles sur des séquences nucléiques ou protéiques, le non déterminisme nous semble très indiqué pour prendre en compte l'accrochage sur des sites actifs, que l'on retrouve dans de nombreux mécanismes biologiques. Considérons par exemple la traduction.

[1]Par exemple, on sait produire un automate canonique minimal en nombre d'états, ce qui n'est pas le cas pour les automates non déterministes.

Une sous-unité du ribosome se fixe sur l'extrémité de l'ARNm, accompagnée de l'ARNt qui porte la méthionine. Le ribosome et la sous-unité se déplacent jusqu'à ce qu'ils parviennent au codon initiateur AUG, codant la méthionine. Une autre sous-unité ribosomale vient alors se fixer sur ce codon et la synthèse démarre. Tant que le codon initiateur n'a pas été rencontré, la "tête de lecture" du ribosome effectue une boucle d'attente sur l'ensemble des caractères. Ceci correspond au deuxième type de modèle présenté dans la figure 4.3.

Maintenant, si l'on admet que les NFA sont adaptés à la représentation des langages observés en biologie, il faut mettre au point des méthodes pour les apprendre de manière automatique à partir des séquences. En effet, on cherche dans tous les cas à minimiser la taille du modèle, et celle-ci est très différente suivant qu'on considère des automates déterministes ou pas.

On retrouve dans l'inférence d'automates non déterministes certaines contraintes de l'apprentissage de grammaires algébriques, en ce sens qu'il faut raffiner la relation d'inclusion sur les automates et qu'il faut choisir des sous classes adaptées permettant la production d'automates canoniques. Le problème de l'inférence d'automates non déterministes avait été traité auparavant par T. Yokomori, en 1994 [225]. Celui-ci propose cependant un algorithme dont l'efficacité repose sur une forte coopération avec l'utilisateur, ce qui n'est pas envisageable dans notre cas. Nous avons essentiellement travaillé pour notre part sur une classe restreinte de NFA, les **automates non ambigus** (UFA). La figure 4.4 illustre le concept d'ambiguïté, relié au nombre maximum d'analyses possibles pour un mot du langage, sur quelques automates du langage a^*. Les UFA conservent une partie des bonnes propriétés des automates déterministes, en n'acceptant qu'une seule analyse pour chaque séquence.

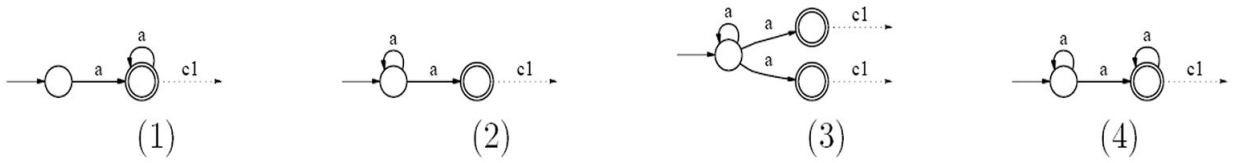


FIG. 4.4 – Quatre représentations du langage a^* : (1) un automate déterministe, (2) un UFA non déterministe, (3) un automate d'ambiguïté 2 et (4) un automate d'ambiguïté infinie

Les principaux résultats obtenus dans la thèse de D. Fredouille affinent la caractérisation de l'espace de recherche des UFA : on exhibe un ordre conférant à cet espace une structure de treillis ainsi que deux opérateurs de parcours associés. Le premier est la fusion déterministe classique ; le second est un nouvel opérateur plus particulièrement adapté au cas non déterministe, la fusion pour désambiguisation, qui permet l'apprentissage d'automates non ambigus. Un des résultats majeurs de cette étude est la proposition d'un algorithme pour l'inférence d'automates non ambigus qui nécessite moins de données que les algorithmes usuels d'inférence d'automates déterministes pour converger.

Une autre ligne de recherche très intéressante d'un point de vue formel a été proposée de façon concurrente à la notre par F. Denis, A. Lemay et A. Terlutte [58]. Elle explore une classe différente d'automates non déterministes, les automates à états résiduels, qui représente une alternative à nos travaux. Il reste à démontrer son pouvoir explicatif dans le cadre de la bioinformatique (la figure 4.3 donne un exemple d'un tel automate).

D. Fredouille a également travaillé sur des méthodes permettant d'introduire, lors de l'inférence, des connaissances a priori sur les séquences, qu'il s'agisse de protéines ou d'ADN : connaissance de zones à ne pas fusionner entre elles (p. ex. typage avec des structures secondaires pour des protéines) ; connaissance de contre-exemples génériques (p. ex. protéines de moins de 10 résidus) ; connaissance de modèles a priori à raffiner ou à généraliser (p. ex. connaissance de l'existence de deux sites actifs dans une protéine ou d'un motif Prosite). La richesse des possibilités dans ce domaine n'a fait qu'être effleurée et nécessite maintenant des études en vraie grandeur.

Cet aller-retour entre théorie et application est illustré également dans une deuxième thèse, celle de A. Leroux, sur le problème de la prise en compte d'un alphabet ordonné en inférence grammaticale [139]. En effet, des séquences comme les macromolécules du vivant ne sont pas formées de lettres indépendantes mais partagent au contraire de nombreuses relations issues des propriétés physico-chimiques communes à certains acides. Il faut donc permettre dans les automates, des transitions sur des sous-ensembles de lettres qui sont autant de généralisations des lettres observées dans chacune des séquences.

Nous avons utilisé le diagramme de Taylor, qui classe les acides aminés suivant leurs propriétés physico-chimiques [213], pour construire de manière automatique, sous forme de treillis, un ordre sur les groupes d'acides aminés. Tout ensemble de propriétés pourrait être pris en compte à la place de celles définies par Taylor. La figure 4.5 donne un aperçu d'une partie du treillis.

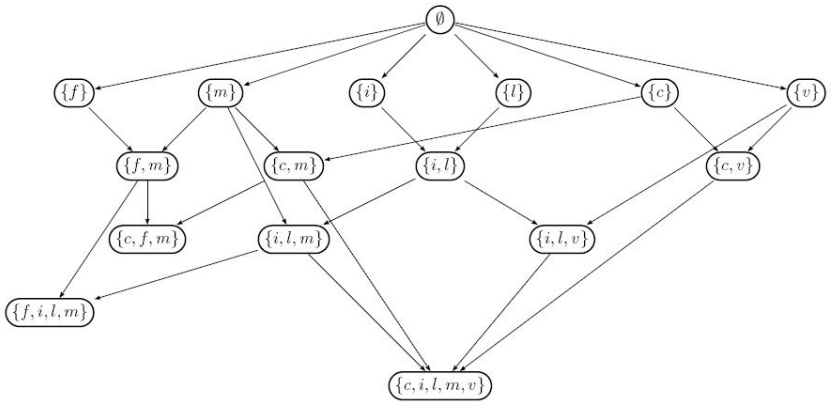


FIG. 4.5 – Partie du treillis sur les ensembles d'acides aminés utilisé pour l'inférence d'automates sur des protéines

Nous avons aussi développé une méthode d'inférence (SDTM, Similarity Driven Transition Merging) [138] qui calcule les meilleurs alignements locaux entre les paires de protéines suivant un score fondé à la fois sur cet ordre et sur les propriétés statistiques de l'ensemble de protéines donné. L'algorithme travaille sur des machines séquentielles que nous appelons *a*utomates de transitions finies, qu'A. Leroux a proposé pour focaliser l'inférence sur la fusion de transitions plutôt que d'états. Ces machines sont proches de celles de Mealy, avec des sorties réduites au statut d'acceptation. Formellement, ces automates sont définis comme suit :

définition 11. (**automate de transitions finies**) Un automate de transitions finies (FTA) est un triplet (Q, Σ, T), où Q est un ensemble fini d'états, Σ est un alphabet fini et T, défini sur $Q \times \Sigma \times 2^{\{i,f\}} \times Q$, est un ensemble fini de transitions. Chaque transition du FTA est un quadruplet $t = (q, l, m, q')$, où $q \in Q$(resp. $q' \in Q$) est l'origine (resp. la destination) de t, $l \in \Sigma$ est le label de la transition et $m \subseteq \{i, f\}$ est la marque de la transition indiquant si t est initiale et/ou finale. Un mot est accepté par un tel automate s'il est accepté par une suite de transitions commençant par une transition initiale et finissant par une transition finale.

On peut définir des versions déterministes et non déterministes de FTA. Nous donnons en figure 4.6 un exemple de FTA déterministe minimal et l'automate déterministe minimal correspondant. Le langage reconnu est l'ensemble des mots de $\{a, b\}^*$ comprenant le mot b et l'ensemble des mots xwy, où x et y sont des lettres et w contient un nombre impair de b.

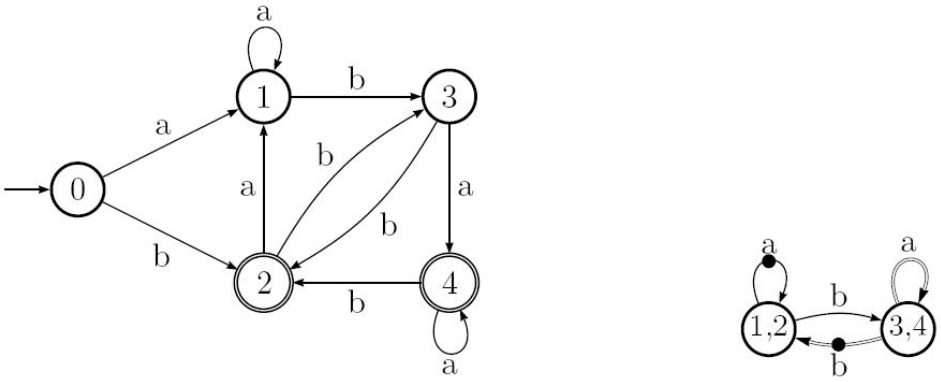


FIG. 4.6 – Exemple de FTA et automate déterministe correspondant. Les transitions initiales sont marquées par un point et les transitions finales sont doublées

L'algorithme SDTM commence par construire le plus grand automate reconnaissant exactement les mots du langage et le généralise par fusions successives des paires de transitions correspondant aux acides aminés appariés dans les alignements sélectionnés.
Par rapport aux méthodes d'inférence par fusion d'états, la différence principale est que l'unité fusionnée est ici une paire d'états adjacents dans l'automate. Les expérimentations ont montré l'intérêt de cette combinaison de méthodes importées de la découverte de motifs et de l'inférence grammaticale, ce qui a permis de lancer une autre thèse en cours sur le sujet (thèse de G. Kerbellec, encadrée par F. Coste).

Enfin, nous avons travaillé avec Y. Mescam sur le problème de l'apprentissage de motifs doubles (dyades), dont les sous-parties sont reliées par un morphisme. Ce type de motifs se rapproche de ceux qu'on rencontre dans les grammaires à variable de chaîne et qui permettent de modéliser des langages contextuels. Nous supposons que si une mutation apparaît sur un des sites de ses motifs doubles dans les protéines, alors soit cette mutation préserve les caractéristiques physico-chimiques nécessaires à l'interaction, soit une mutation compensatrice apparaît sur l'autre site de façon à restaurer ces propriétés [81]. L'inférence de telles expressions ne peut raisonnablement pas être réalisée de manière exacte. Nous avons essayé des algorithmes de type

évolutionniste, en coopération avec le SIB à Genève (R. Gras et D. Hernandez). Le problème principal était la localisation des sites co-variant au cours du temps. Ce processus conduit à des corrélations statistiquement mesurables entre sites et nous avons proposé une nouvelle mesure pour estimer ces corrélations. Nous nous sommes servis de l'approche utilisée dans l'algorithme MoDEL [104], qui recherche le meilleur candidat avec une approche métaheuristique à travers l'espace de tous les motifs possibles. Nous avons étendu cet algorithme de façon à autoriser la localisation de sites multiples, et avons obtenu des performances similaires à celles du "Gibbs motif sampler". Cette méthode a montré des résultats satisfaisants sur des séquences artificielles sur lesquelles étaient implantées des caractéristiques de mutations "naturelles" [94, 95]. Nous avons pu par contre découvrir de nouvelles relations sur des séquences de diverses familles de protéines.

4.5 Conclusion

La recherche en inférence grammaticale est exigeante. La confrontation entre théories et pratique y est particulièrement difficile.

Nous avons appris à progressivement baisser nos exigences dans ce domaine. Cependant, les premiers résultats sont là. Les travaux de F. Coste et G. Kerbellec ont abouti à la première méthode à notre connaissance à être applicable sur des séquences réelles de protéines [49]. Protomata est disponible pour les laboratoires de biologie via un site web sur la plate-forme GenOuest (http ://protomata-learner.genouest.org). Son premier avantage par rapport aux motifs traditionnels de Prosite est d'autoriser des alternatives variées à l'intérieur des séquences dans leur composition en sous-motifs.
Il reste à exploiter pleinement le potentiel de l'outil sur la modélisation de familles de protéines, mais les premiers résultats sont d'ores et déjà très encourageants.

Nous voudrions terminer en soulignant trois points de perspective qui nous semblent importants par rapport à l'inférence grammaticale.

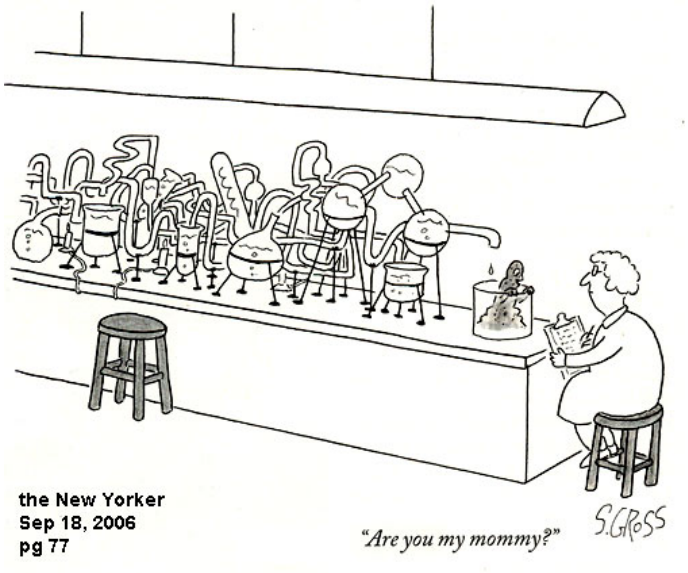
- Bien qu'étant une méthode d'apprentissage supervisée, c'est-à-dire où les instances d'apprentissage sont étiquetées, elle délivre des **résultats de nature non supervisée**. En effet, une grammaire ou un automate possède d'une manière générale plusieurs non-terminaux ou états d'acceptation. Il est donc possible non seulement de reconnaître une famille donnée, mais également de distinguer a posteriori des sous-familles à l'intérieur de celle-ci. Ce point me semblent particulièrement intéressant dans le contexte de la bioinformatique et méritera d'être exploité.
- Le problème du passage à l'échelle des méthodes d'inférence grammaticale sur des séquences longues (de l'ordre du millier de lettres) reste critique à la fois du point de vue algorithmique et du point de vue des résultats. Un bon point de départ face à cette difficulté est de **bien séparer le problème de la localisation de sites particulièrement actifs de celui de la reconnaissance globale des enchaînements de ces sites**. On rejoint ici une problématique commune à l'analyse syntaxique, qui exige en premier lieu une bonne analyse lexicale. L'objectif reste encore largement à atteindre dans le cadre de l'inférence grammaticale. La solution viendra probablement d'un bon équilibre entre recherche de motifs et inférence grammaticale.

Du point de vue des résultats, des expériences préliminaires que nous avons menées sur des séquences longues et/ou nombreuses conduisent généralement à de très grands automates qu'il faut pouvoir simplifier. Parmi les actions de post-traitement qui nous semblent les plus pertinentes, il serait souhaitable d'étudier la "probabilisation" des transitions des automates et les fusions d'états supplémentaires que permettrait ce passage.

– Enfin, le plein emploi des méthodes d'inférence grammaticale suppose des progrès décisifs en **apprentissage de grammaires algébriques**. Nous avons montré deux pistes importantes dans cette optique. La première consiste à choisir une représentation n'hésitant pas à "traverser la hiérarchie de Chomsky" (n'incluant pas tous les langages réguliers ni tous les langages algébriques) pour modéliser des phénomènes non réguliers. Des systèmes formels type systèmes de réécriture peuvent être de bons candidats pour cela. La deuxième piste consiste à importer et intégrer des résultats en provenance d'algorithmes de compression grammaticale.

Chapitre 5

Introduire une assistance à l'expérimentation par le raisonnement automatique



Nous terminons ce document par nos perspectives à moyen terme, dans une nouvelle aventure scientifique qui cherche à favoriser encore davantage la confrontation des modèles du chercheur en biologie avec les observations sur le vivant.

Modéliser les structures sur les séquences d'acides nucléiques ou protéiques est indispensable pour comprendre comment l'information est gérée à l'intérieur de la cellule. Cependant, le biologiste raisonne sur un univers beaucoup plus large qui comprend toutes les interactions biochimiques à l'intérieur et à l'extérieur de la cellule, ainsi que l'adaptation à l'environnement. Avec

la progression de l'instrumentation et l'afflux des données, les modèles qu'il conçoit deviennent extrêmement complexes. Il faut là aussi lui fournir les outils pour que ces modèles puissent faire l'objet d'études automatisées sur leurs propriétés intrinsèques, qu'ils puissent être testés expérimentalement de façon rigoureuse, et enfin qu'ils puissent être précisés ou amendés en fonction des résultats.

La biologie des systèmes est devenue une préoccupation centrale de la communauté internationale en bioinformatique. Deux questions sont au cœur des recherches : "comment intégrer l'ensemble des sources de connaissance sur le vivant de manière cohérente et en prenant en compte les aspects multi- échelles ? " et " comment simuler les mécanismes du vivant ? ".

Notre propre expérience du domaine nous pousse à considérer deux points comme prioritaires :

– **Le bioinformaticien doit intervenir avec le biologiste pour aider à la conception des expérimentations** qui permettront de faire progresser la connaissance d'un mécanisme biologique.
  En effet, les questions d'identification de système qui se posent du fait de l'arrivée massive de données ne sont pas naturelles pour des expérimentateurs formés à une démarche réductionniste. Il est par exemple intéressant de faire varier plusieurs variables à la fois lors de l'analyse du comportement d'un système, ce qui n'est actuellement pas du tout une évidence dans les laboratoires de biologie.
  De plus, le coût et la reproductibilité des expérimentations restent des facteurs bloquants, même si les moyens technologiques ont beaucoup progressé. La technologie est en avance de phase sur la modélisation, et le biologiste passe maintenant plus de temps à chercher la maîtrise des machines qu'à optimiser sa démarche expérimentale (exemple des puces à ADN). Pour renverser cette tendance, il faut penser dès l'origine l'adaptation du dispositif expérimental en fonction des objectifs et non l'inverse. Ceci suppose de fédérer encore plus largement les compétences dans des disciplines variées (électronique, informatique, biologie...) pour la conception de ce dispositif.
– **Le bioinformaticien doit mettre en place des méthodes qui permettront au biologiste d'exploiter réellement les bases de connaissances informatisées de son domaine**. L'intégration des données s'accélère et mène à de véritables recueils de connaissances qui ne peuvent plus être exploités manuellement. Ainsi on vient de disposer d'une base de données répertoriant l'ensemble des métabolites, éléments nutritifs et médicaments pouvant être présents dans le corps humain [66] (plus de 7000 entrées) et, parallèlement, d'une liste complète des réactions métaboliques connues et publiées depuis 50 ans pour l'homme [223] (BiGG -Biochimiquement, Génétiquement et Génomiquement structuré- , plus de 3000 réactions).
  Il ne s'agit pas simplement d'interroger ces bases par mots-clés ou même par requête SQL, il faut une véritable interface capable d'assister le chercheur dans son raisonnement. De façon contradictoire, il nous semble clair que la conception de simulateurs doit être réduite à des mécanismes soigneusement sélectionnés car la biologie des systèmes exhibe une complexité globale de fonctionnement qui n'est absolument pas observable par la technologie actuelle, et probablement encore pour longtemps. Une voie raisonnable semble se dessiner avec la biologie synthétique où on cherche à reconstruire des portions limitées d'un système vivant. Le problème bioinformatique est alors d'isoler ces sous-systèmes puis d'expliquer

les observations effectuées en reconstruisant éventuellement des chemins métaboliques ou de signalisation manquants.

5.1 Laboratoires robotisés et laboratoires sur puce

L'expérimentation en biologie est un compromis entre des dimensions incompatibles : le coût et la pertinence des observations d'une part et leur reproductibilité et exhaustivité d'autre part. On disposait jusqu'il y a peu de temps de trois échelles d'expérimentation :

- Le *in vivo*, ultime confrontation des modèles à la réalité biologique, présente les meilleures garanties de pertinence et le coût le plus élevé et il est très difficile de mener des expériences systématiques et au comportement parfaitement maîtrisé.
- Le *in vitro* cherche à réduire les paramètres d'expérimentation en isolant des sous-systèmes hors organisme, dans des conditions physiologiques imposées. On peut envisager plus d'expériences à moindre coût, on accède à une meilleure reproductibilité (plus ou moins...) des expériences. Le prix à payer est un certain éloignement des conditions réelles, mais qui reste dans des limites raisonnables.
- Enfin, plus récemment, le *in silico*[1] a envahi les laboratoires de biologie. Il suffit de s'y promener un peu pour constater à quel point les étudiants en thèse de biologie augmente la part de travail sur ordinateur par rapport à celui sur paillasse. Les limites de l'expérimentation sont alors uniquement celles du calcul pour certains modèles complexes, et la reproductibilité est parfaite. Par contre, on peut passer complètement à côté de la réalité biologique et il faut revenir régulièrement à l'expérimentation in vitro puis in vivo pour valider les hypothèses. Comme nous l'avons illustré dans ce document, le 'in silico' ne se réduit pas à la simulation. La conception de modèles joue un rôle clé dans la conception de futures expériences.

Du point de vue des dispositifs technologiques récents, il se développe une nouvelle révolution de l'approche expérimentale avec le champ de conception de systèmes appelés *micro-systèmes d'analyse totale (TAS, μTAS, Miniaturized Total Chemical Analysis System ou Micro Total Analysis System) appelés plus médiatiquement laboratoires sur puce (LOC ou Lab On Chip)*. On trouvera des revues du domaine dans [115, 187, 16, 222, 151, 105]. Il existe également depuis 2001 un journal spécialisé sur le thème des laboratoires sur puce (Lab on Chip).

Durant les années 1990, on a assisté à une adaptation des techniques de microfabrication de circuit pour la manipulation de molécules biologiques. L'échelle de réalisation reste confortable par rapport à d'autres dispositifs nanométriques. Ces systèmes devraient permettre de développer rapidement à la fois des systèmes d'analyse à très haut débit peu coûteux et puissants, et des systèmes simples d'usage à des fins applicatives dans les domaines de la santé et des biotechnologies.

L'éventail des fonctionnalités d'expérimentation que permettent ces technologies s'enrichit régulièrement. On peut par exemple distribuer un produit en parallèle dans plusieurs canaux de manière électriquement contrôlée à partir d'un canal de chargement ; mélanger des produits ; diffu-

[1] Terme forgé à l'origine par A. Danchin ?

ser en spray des molécules ; séparer des produits par électrophorèse et chromatographie gazeuse ; aspirer des protéines ou pomper des gaz dans un milieu ; contrôler les paramètres de température et de pression... La tendance est maintenant à l'intégration de différentes fonctions en vue d'une tâche donnée [71].
Les avantages décisifs que procurent ces microtechnologies sont l'excellent niveau de contrôle des conditions expérimentales (les manipulations sont extrêmement réduites et l'environnement clos peut être très bien contrôlé), l'abaissement des coûts (les quantités utilisées sont minimes), et la possibilité de mener des expérimentations intensives. On surpasse ainsi largement les possibilités standard des expérimentations in vivo. Comme conséquence directe, il en résulte cependant une avalanche potentielle de mises au point d'expériences et de résultats à analyser pour lesquels on dispose encore de très peu de moyens et qui limitent de fait l'intérêt actuel de ces systèmes.

Un projet pionnier, baptisé "Adam" et publié dans le journal Nature en 2004 [128], a ouvert une voie de recherche fondamentale vis-à-vis de ce problème du couplage entre expérimentateur et expérimentation. Le " robot-scientist" Adam est une plate-forme couplant un robot de manipulation de liquides, un plateau avec des micros puits (microtitre) et un **système de contrôle et de raisonnement hypothético-déductif**. Nous ne sommes pas ici dans un micro-système mais dans un système d'expérimentation plus traditionnel, largement robotisé (et financé) par Caliper. Les coupelles du plateau sont transférées manuellement dans des incubateurs lors des expérimentations. Le système a été appliqué à la compréhension de la fonction des gènes impliqués dans la synthèse des acides aminés aromatiques, sur la levure.

La nouveauté fondamentale qu'introduit le système Adam est bien sûr le module de raisonnement. Initialement, le laboratoire dispose d'une base de données de chemins métaboliques impliqués dans la synthèse d'acides aminés. Il dispose également d'un ensemble de souches mutantes de la levure, dont on a enlevé un gène et qui ne peuvent plus croître sur un milieu standard. Les expérimentations restent simples et consistent à faire croître une souche sur un milieu défini (une base et un ou deux métabolites) et à mesurer la concentration de celle-ci. Grâce à une représentation logique des connaissances et un raisonnement abductif puis déductif, le système tente d'optimiser le coût et le nombre d'expérimentations nécessaires pour inférer la relation entre le gène absent d'une souche et l'enzyme correspondante. Les auteurs montrent que le système améliore les performances de l'expérimentation par rapport à une stratégie de choix aléatoire ou de moindre coût.

Notre propre projet vise à rechercher au sein d'un même projet l'intégration de capacités de raisonnement et de capacités d'expérimentation sur micro-laboratoire. Il est probable que de tels projets vont se multiplier dans les années à venir. Les problèmes scientifiques ne manquent pas pour obtenir une chaîne complète, mais il faudra également aux équipes qui s'y attaqueront une composante essentielle : la mise en commun de compétences de multiples disciplines.

5.2 Basic Lab, un projet de laboratoire sur puce intégrant un module de raisonnement

Description détaillée du projet :
Notre idée est de mettre en oeuvre à une échelle micro ou nanométrique une chaîne roboti-

sée complète et généraliste d'expérimentation scientifique au niveau moléculaire sur les cellules, incluant en boucle les étapes de sélection d'hypothèses à partir des connaissances et des observations, de conception d'une expérimentation permettant de tester l'hypothèse choisie, de mise en oeuvre de l'expérimentation, et enfin de recueil des résultats d'expérimentation et de mise à jour de la base de connaissances.

Notre projet Basic Lab repose sur une coopération entre des biologistes (Inserm U522, UMR CNRS 6026), des électroniciens (IETR, microélectronique) des spécialistes de microfluidique Biomis, ENS Cachan Antenne de Bretagne) et bien sûr, des informaticiens / mathématiciens. Il propose de réfléchir sur trois points majeurs :

- La réalisation d'une boucle d'expérimentation robotisée complète ;
- La conception d'une machine programmable d'expérimentation biochimique sur les cellules, suffisamment générique pour traiter des tâches scientifiques variées ;
- l'introduction d'un système de raisonnement complet intégrant une base de connaissances importante sur les cellules et leur métabolisme, une logique expressive permettant la description partielle de systèmes et de leur dynamique, un démonstrateur, un diagnostiqueur et un planificateur permettant de vérifier des hypothèses, d'en générer de nouvelles à partir des comportements observés à l'intérieur de la cellule puis de générer les expérimentations complémentaires permettant de vérifier ces hypothèses.

L'idée générale est de permettre la circulation de fluides à échelle nanométrique vers de nombreuses chambres de réactions en parallèle. Du point de vue des problèmes de recherche associés, on distingue la partie biologique, où il s'agit d'expérimenter un modèle réaliste de tissu in vitro, la partie technologique, où il s'agit de mettre au point la chaîne de dispositifs physiques permettant l'expérimentation, et la partie informatique où il s'agit de mettre au point la chaîne de traitement de données permettant de produire et de raisonner sur des connaissances biochimiques.

En terme d'applications, ce que nous visons est la possibilité d'expérimentation in vitro sur un type cellulaire défini, avec des conditions proches du in vivo et avec un impact fort en termes de retombée sur la santé. Avec ces véritables " cobayes sur puce ", on réduit fortement la nécessité, le temps et le coût de l'expérimentation animale. Le candidat idéal pour les expérimentations sur puce doit correspondre à un ensemble de critères assez difficiles à réunir en pratique :

- Possibilité de conservation de lignées cellulaires sur quelques jours ;
- Comportement proche du " in vivo " ;
- Intérêt thérapeutique ;
- Disponibilité de connaissances théoriques et expérimentales formalisables ;
- Intérêt pratique d'un grand nombre d'expérimentations ;

Nous nous proposons de travailler en toxicologie sur des lignées d'hépatocytes humains, ainsi que sur des bactéries soumises à un stress oxydatif.

Issu d'une équipe de représentation des connaissances, j'ai travaillé à mes débuts dans le domaine logique et je retrouve dans ce projet le moyen de mettre en pratique cet acquis. L'automatisation du raisonnement est au coeur des possibilités de souplesse du dispositif d'expérimentation et nous pouvons bénéficier maintenant de progrès notables dans ce domaine par rapport au raison-

nement dit “de sens commun”, typique de l’environnement expérimental.
Je suggère de privilégier une **formalisation logique par logiques d’actions** [88], car elle permet d’aborder à la fois des aspects déductifs (exploration), abductifs (diagnostic) et de planification (expérimentation). Plus précisément, le système doit intégrer ces capacités sophistiquées pour couvrir la boucle expérimentale : la déduction pour inférer l’ensemble des conséquences des conditions imposées et des observations effectuées pendant l’expérimentation, l’abduction pour inférer les hypothèses manquantes pour expliquer ces conséquences à partir de la base courante de connaissances, et enfin l’induction de plans d’expérimentation pour discriminer de manière optimale parmi les hypothèses concurrentes celle qui sera testée.
La représentation logique a été également utilisée dans l’expérimentation du Robot Scientist, pour modéliser et inférer des réseaux métaboliques sur la levure [186], mais dans un cadre plus traditionnel de programmation logique. Les expérimentations y restaient relativement simples et le choix d’une hypothèse entraînait directement celui de l’expérience . Dans le cadre que nous définissons d’une machine générique d’expérimentation, les problèmes de planification seront plus complexes, car les expériences incluront plusieurs étapes. Le projet Inria Contraintes (F. Fages) propose un cadre logique plus expressif, permettant en particulier d’introduire des notions temporelles [76] et nous allons bénéficier de l’expérience acquise. Nous commençons par traduire dans ces formalismes logiques les banques de données spécialisées sur le métabolisme , qui ne font pour le moment que l’objet de requêtes limitées. Les logiques d’actions ont l’avantage d’être associées à des solveurs extrêmement efficaces, dans le cadre de la " programmation par ensembles réponses " (Answer Set Programming [143]), exploitant pleinement les techniques des solveurs SAT et des solveurs de contraintes. Ceci rend possible l’étude de systèmes en vraie grandeur. L’atout majeur de l’ASP est de pouvoir gérer naturellement une connaissance incomplète en évolution dans un cadre non monotone très bien défini (les logiques non monotones permettent de remettre en cause les théories échafaudées avec un ensemble d’observations donné lorsqu’une observation supplémentaire les contredit). La programmation logique utilise une négation par absence de dérivation possible, alors que l’ASP permet d’interpréter la négation plus naturellement comme la consistance du caractère faux d’un fait. Nous travaillons sur ce point en étroite collaboration avec l’université de Potsdam (T. Schaub [87]). Une première thèse a montré les capacités du formalisme en prédiction [220] et nous avons commencé l’encadrement d’une thèse en cotutelle sur le sujet (T. Hénin). Un premier travail débouchant sur une thèse a été soutenu sur la production d’hypothèses sur les réseaux de signalisation à Arizona State Univ. montrant l’intérêt et la faisabilité de l’approche [218, 19]. Sans vouloir me lancer dans une prospective hasardeuse, cela ne m’étonnerais pas de voir la communauté scientifique du domaine croître rapidement dans les dix ans à venir. La Biologie sera, n’en doutons pas, la science du XXIème siècle !

Bibliographie

[1] Abouelhoda, Kurtz, and Ohlebusch. The enhanced suffix array and its applications to genome analysis. In *WABI : International Workshop on Algorithms in Bioinformatics, WABI, LNCS*, 2002.

[2] M.I. Abouelhoda, S. Kurtz, and E. Ohlebusch. Enhanced suffix arrays and applications. In S. Aluru, editor, *Handbook on Computational Molecular Biology*, pages 1–27. Chapman and Hall/CRC, 2006.

[3] Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1) :53–86, 2004.

[4] G. Achaz, É. Coissac, A. Viari, and P. Netter. Analysis of intrachromosomal duplications in yeast *Saccharomyces cerevisiae* : a possible model for their origin. *Mol. Biol. Evol.*, 17(8) :1268–75, 2000.

[5] G. Achaz, E. P. Rocha, P. Netter, and É. Coissac. Origin and fate of repeats in bacteria. *Nucl. Acids Res.*, 30(13) :2987–94, 2002.

[6] Alfred V. Aho. *Algorithms for finding patterns in strings*, pages 255–300. MIT Press, Cambridge, MA, USA, 1990.

[7] Alfred V. Aho and Jeffrey D. Ullman. *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1972.

[8] C. M. Ajo-Franklin, D. A. Drubin, J. A. Eskin, E. P. Gee, D. Landgraf, I. Phillips, and P. A. Silver. Rational design of memory in eukaryotic cells. *Genes Dev*, 21(18) :2271–2276, September 2007.

[9] Henk Alblas. Introduction to attributed grammars. In Henk Alblas and Borivoj Melichar, editors, *Attribute Grammars, Applications and Systems, International Summer School SAGA, Prague, June 4-13, 1991, Proceedings*, volume 545 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1991.

[10] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3) :403–410, October 1990.

[11] Dana Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39(3) :337–350, 1978.

[12] Dana Angluin. Finding patterns common to a set of strings (extended abstract). In *STOC '79 : Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 130–141, New York, NY, USA, 1979. ACM Press.

[13] A. Apostolico. The myriad virtues of suffix trees. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Words*, volume 12 of *NATO Advanced Science Institutes, Series F*, pages 85–96. Springer-Verlag, Berlin, 1985.

[14] Alberto Apostolico and Stefano Lonardi. Compression of biological sequences by greedy off-line textual substitution. In *Data Compression Conference*, pages 143–152, 2000.

[15] M. R. Atkinson, M. A. Savageau, J. T. Myers, and A. J. Ninfa. Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in escherichia coli. *Cell*, 113(5) :597–607, May 2003.

[16] P.-A. Auroux, D. Iossifidis, D. Reyes, and A. Manz. Micro total analysis systems. 2. analytical standard operations and applications. *Analytic Chemistry*, 74 :2637–2652, 2002.

[17] R. K. Azad, J. S. Rao, W. Li, and R. Ramaswamy. Simplifying the mosaic description of dna sequences. *Physical Review*, E66(031913), 2002.

[18] Albert-László Barabási. *Linked : The New Science of Networks*. Perseus Books Group, May 2002.

[19] C. Baral, K. Chancellor, N. Tran, N. L. Tran, A. Joy, and M. Berens. A knowledge based approach for representing and reasoning about signaling networks. *Bioinformatics*, 20 Suppl 1, August 2004.

[20] Catherine Belleannée and Jacques Nicolas. Logol : Modelling evolving sequence families through a dedicated constrained string language. Research Report 6350, INRIA, 11 2007.

[21] Frédéric Benhamou and Touraïvane. Prolog iv : langage et algorithmes. In Jean-Jacques Chabrier, editor, *JFPLC*, pages 51–64, 1995.

[22] P. Bernaola-Galvn, R. Romn-Roldn, and J.L. Oliver. Compositional segmentation and long-range fractal correlations in dna sequences. *Physical Review*, E53 :5181–5189, 1996.

[23] P. Bertrand and E. Diday. Une généralisation des arbres hiérarchiques : les représentations pyramidales. *Revue de Statistique Appliquée*, 38(3) :53–78, 1990.

[24] Doron Betel and Christopher W. V. Hogue. Kangaroo - a pattern-matching program for biological sequences. *BMC Bioinformatics*, 3 :20, 2002.

[25] B Billoud, M Kontic, and A Viari. Palingol : a declarative programming language to describe nucleic acids' secondary structures and to scan sequence database. *Nucleic Acids Res*, 24(8), April 15 1996.

[26] Anna Blenda, Jodi Scheffler, Brian Scheffler, Michael Palmer, Jean-Marc Lacape, John Z. Yu, Christopher Jesudurai, Sook Jung, Sriram Muthukumar, Preetham Yellambalase, Stephen Ficklin, Margaret Staton, Robert Eshelman, Mauricio Ulloa, Sukumar Saha, Ben Burr, Shaolin Liu, Tianzhen Zhang, Deqiu Fang, Alan Pepper, Siva Kumpatla, John Jacobs, Jeff Tomkins, Roy Cantrell, and Dorrie Main. Cmd : a cotton microsatellite database resource for gossypiumgenomics. *BMC Genomics*, 7 :132, 2006.

[27] D. Blount and D. Grogan. New insertion sequences of *Sulfolobus* : functional properties and implications for genome evolution in hyperthermophilic archaea. *Mol. Microbiol.*, 55(1) :312–25, 2005.

[28] Valentina Boeva, Mireille Regnier, Dmitri Papatsenko, and Vsevolod Makeev. Short fuzzy tandem repeats in genomic sequences, identification, and possible role in regulation of gene expression. *Bioinformatics*, 22(6) :676–684, 2006.

[29] Alvis Brazma and David Gilbert. A pattern language for molecular biology. Technical Report 11, City University, London, 1995.

[30] Volker Brendel, Jacques S. Beckmann, and Edward N. Trifonov. Linguistics of nucleotide sequences : Morphology and comparison of vocabularies. *Journal of Biomolecular Structure and Dynamics*, 4(1) :11–21, 1986.

[31] Volker Brendel and H.G. Busse. Genome structure described by formal languages. *Nucleic Acids Research*, 12(5) :2561–2568, February 1984.

[32] G.S. Brodal, R.B. Lyngs, C.N.S. Pedersen, and J. Stoye. Finding maximal pairs with bounded gap. In *CPM*, pages 134–149, 1999.

[33] S. Burbeck and K. E. Jordan. An assessment of the role of computing in systems biology. *IBM J. Res. Dev.*, 50(6) :529–543, 2006.

[34] Chris Burge. Bioinformaticists will be busy bees. *Genome Technology*, 17, January 2002.

[35] Robert D. Cameron. Source encoding using syntactic information source models. *IEEE Transactions on Information Theory*, 34(4) :843–850, 1988.

[36] Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, April Rasala, Amit Sahai, and abhi shelat. Approximating the smallest grammar : Kolmogorov complexity in natural models. In *STOC '02 : Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 792–801, New York, NY, USA, 2002. ACM.

[37] Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai, and Abhi Shelat. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7) :2554–2576, 2005.

[38] B. Charlesworth, P. Sniegowski, and W. Stephan. The evolutionary dynamics of repetitive DNA in eukaryotes. *Nature*, 371(6494) :215–220, 1994.

[39] Gilbert Chauvet. *La vie dans la matière - le role de l'espace en biologie*. Flammarion, 1998.

[40] Churchill. Stochastic models for heterogeneous DNA sequences. *BMB : Bulletin of Mathematical Biology*, 51(1) :79–94, 1989.

[41] B. Clift, D. Haussler, R. McConnell, T. D. Schneider, and G. D. Stormo. Sequence landscapes. *Nucl. Acids Res.*, 14(1) :141–158, 1986.

[42] Jacques Cohen. A view of the origins and development of prolog. *Commun. ACM*, 31(1) :26–36, 1988.

[43] Jacques Cohen. Bioinformatics - an introduction for computer scientists. *ACM Computing Surveys*, 36(2) :122–158, 2004.

[44] Julio Collado-Vides. A transformational-grammar approach to the study of the regulation of gene expression. *Journal of Theoretical Biology*, 136 :403–425, 1989.

[45] A. Colmerauer. Les Systèmes Q ou un Formalisme pour Analyser et Synthétiser des Phrases sur Ordinateur. Technical report, University of Montreal, 1970. (Internal Publication No. 43).

[46] Alain Colmerauer and Philippe Roussel. *The birth of Prolog*, pages 331–367. ACM Press, New York, NY, USA, 1996.

[47] International HapMap Consortium. A second generation human haplotype map of over 3.1 million snps. *Nature*, 449(7164) :851–861, October 2007.

[48] François Coste and Jacques Nicolas. Inference of finite automata : Reducing the search space with an ordering of pairs of states. In Claire Nedellec and Céline Rouveirol, editors, *ECML*, volume 1398 of *Lecture Notes in Computer Science*, pages 37–42. Springer, 1998.

[49] François Coste and Goulven Kerbellec. Learning automata on protein sequences. In *Jobim'06*, Bordeaux, 2006.

[50] Francois Coste and Jacques Nicolas. Regular inference as a graph coloring problem. In *ICML*, December 1997.

[51] François Coste and Jacques Nicolas. How considering incompatible state mergings may reduce the DFA induction search tree. In Vasant Honavar and Giora Slutzki, editors, *Proceedings of the 4th International Colloquium on Grammatical Inference (ICGI-98)*, volume 1433 of *LNAI*, pages 199–210, Berlin, jul 1998. Springer.

[52] François Coste. *Apprentissage d'automates classifieurs en inférence grammaticale*. PhD thesis, Université de Rennes I, janv. 2000.

[53] François Coste and Jacques Nicolas. L'inférence grammaticale régulière vue comme un problème de coloriage et propagation de contraintes sur un graphe. In *3èmes Journées Nationales sur la Résolution Pratique de de Problèmes NP-Complets (JNPC'97)*, 1997.

[54] Maxime Crochemore and Renaud Vérin. Zones of low entropy in genomic sequences. *Computers & Chemistry*, 23(3-4) :275–282, 1999.

[55] Dabrowski and Plandowski. On word equations in one variable. In *MFCS : Symposium on Mathematical Foundations of Computer Science*, 2002.

[56] Antoine Danchin. *La barque de Delphes : ce que révèle le texte des génomes*. Impressum Paris : Editions Odile Jacob, 1998.

[57] Samuels DC, Boys RJ, Henderson DA, and Chinnery PF. A compositional segmentation of the human mitochondrial genome is related to heterogeneities in the guanine mutation rate. *Nucleic Acids Res*, 31 :6043–52, Oct 2003.

[58] François Denis, Aurélien Lemay, and Alain Terlutte. Learning regular languages using rfsas. *Theor. Comput. Sci.*, 313(2) :267–294, 2004.

[59] J. Devereux, P. Haeberli, and O. Smithies. A comprehensive set of sequence analysis programs for the VAX. *Nucleic Acids Research*, 12(1) :387–395, 1984.

[60] Juan J. Diaz-Mejia, Ernesto Perez-Rueda, and Lorenzo Segovia. Metabolism evolution by gene duplication : a network perspective. *Genome Biology*, 8 :R26+, February 2007.

[61] Volker Diekert. Makanin's Algorithm. In M. Lothaire, editor, *Algebraic Combinatorics on Words*, volume 90 of *Encyclopedia of Mathematics and its Applications*, chapter 12, pages 342–390. Cambridge University Press, 2002.

[62] T. Dobzhansky. Nothing in biology makes sense except in the light of evolution. *American Biology Teacher*, 35 :125–129, March 1973.

[63] Shang. Dong and David. B. Searls. Gene structure prediction by linguistic methods. *Genomics*, 23(3) :540–551, 1994.

[64] Robin Dowell and Sean Eddy. Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinformatics*, 5, July 03 2004.

[65] Robin D Dowell and Sean R Eddy. Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics*, 7 :400, September 04 2006.

[66] Natalie C C. Duarte, Scott A A. Becker, Neema Jamshidi, Ines Thiele, Monica L L. Mo, Thuy D D. Vo, Rohith Srivas, and Bernhard O O. Palsson. Global reconstruction of the human metabolic network based on genomic and bibliomic data. *Proc Natl Acad Sci U S A*, January 2007.

[67] Pierre Dupont, Laurent Miclet, and Enrique Vidal. What is the search space of the regular inference ? In Rafael C. Carrasco and José Oncina, editors, *ICGI*, volume 862 of *Lecture Notes in Computer Science*, pages 25–37. Springer, 1994.

[68] P. Durand, F. Mahé, A.-S. Valin, and J. Nicolas. Pyramid diagram : visualizing the organization of repetitive sequences in genomes. Research Report 5798, INRIA, December 2005.

[69] P. Durand, F. Mahe, A.-S. Valin, and J. Nicolas. Browsing repeats in genomes : Pygram and an application to non-coding region analysis. *BMC Bioinformatics*, 7 :477, 2006.

[70] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme J. Mitchison. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[71] C. J. Easley, J. M. Karlinsey, J. M. Bienvenue, L. A. Legendre, M. G. Roper, S. H. Feldman, M. A. Hughes, E. L. Hewlett, T. J. Merkel, J. P. Ferrance, and J. P. Landers. A fully integrated microfluidic genetic analysis system with sample-in-answer-out capability. *Proc Natl Acad Sci U S A*, 103(51) :19272–19277, December 2006.

[72] S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9) :755–763, 1998.

[73] R. C. Edgar and E. W. Myers. Piler : identification and classification of genomic repeats. *Bioinformatics*, 21(suppl 1) :i152–158, 2005.

[74] Ingvar Eidhammer, Inge Jonassen, Svenn Helge Grindhaug, David Gilbert, and Madu Ratnayake. A constraint based structure description language for biosequences. *Constraints*, 6(2/3) :173–200, 2001.

[75] Sven Ekdahl and Erik L. L. Sonnhammer. Chromowheel : a new spin on eukaryotic chromosome visualization. *Bioinformatics*, 20(4) :576–577, 2004.

[76] François Fages. Symbolic model-checking for biochemical systems. In Catuscia Palamidessi, editor, *ICLP*, volume 2916 of *Lecture Notes in Computer Science*, page 102. Springer, 2003.

[77] Martin Farach-Colton, Gad M. Landau, Süleyman Cenk Sahinalp, and Dekel Tsur. Optimal spaced seeds for faster approximate string matching. *J. Comput. Syst. Sci.*, 73(7) :1035–1044, 2007.

[78] L. F. Fass. Learning context-free languages from their structured sentences. *SIGACT News*, 15 :24–35, 1983.

[79] J. W. Fickett. Recognition of protein coding regions in DNA sequences. *Nucleic Acids Research*, 10(17) :5303–5318, 1982.

[80] W. Fiers, R. Contreras, F. Duerinck, G. Haegeman, D. Iserentant, J. Merregaert, W. Min Jou, F. Molemans, A. Raeymaekers, A. van den Berghe, G. Volckaert, and M. Ysebaert. Complete nucleotide sequence of bacteriophage MS2 RNA : primary and secondary structure of the replicase gene. *Nature*, 260 :500–507, apr 1976.

[81] Gabriele Ausiello Florencio Pazos, Manuela Helmer Citterich and Alfonso Valencia. Correlated mutations contain information about protein-protein interaction. *Journal of Molecular Biology*, 271(4) :511–523, 1997.

[82] Evelyn Fox Keller. *Le siècle du gène*. Gallimard, 2003.

[83] Daniel Fredouille. *Inférence d'automates finis non déterministes par gestion de l'ambiguïté, en vue d'applications en bioinformatique.* PhD thesis, Université de Rennes I, oct. 2003.

[84] R. Friedman and A. L. Hughes. Gene duplication and the structure of eukaryotic genomes. *Genome Res.*, 11(3) :373–81, 2001.

[85] B. Ganter and R. Wille. *Formal Concept Analysis, Mathematical foundations*. Springer, 1997.

[86] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767) :339–342, January 2000.

[87] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. Conflict-driven answer set solving. In Manuela M. Veloso, editor, *IJCAI*, pages 386–, 2007.

[88] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *J. Log. Program.*, 17(2/3&4) :301–321, 1993.

[89] Marian Gheorghe and Victor Mitrana. A formal language-based approach in biology. *Comparative and Functional Genomics*, 5(1) :91–94, 2004.

[90] A. J. Gibbs and G. A. McIntyre. The diagram, a method for comparing sequences. its use with amino acid and nucleotide sequences. *Eur J Biochem*, 16(1) :1–11, 1970.

[91] Robert Giegerich and Stefan Kurtz. From ukkonen to mccreight and weiner : A unifying view of linear-time suffix tree construction. *Algorithmica*, 19(3) :331–353, 1997.

[92] Aristides Gionis and Heikki Mannila. Finding recurrent sources in sequences. In *RECOMB*, pages 123–130, 2003.

[93] Jean-Yves Giordano. *Inférence de grammaires algébriques.* PhD thesis, Université de Rennes I, janv. 1995.

[94] R. Gras, D. Hernández, P. Hernandez, N. Zangge, Y. Mescam, J. Frey, O. Martin, J. Nicolas, and R. D. Appel. Cooperative metaheuristics for exploring proteomic data. *Artif. Intell. Rev*, 20(1-2) :95–120, 2003.

[95] R. Gras, D. Hernandez, P. Hernandez, N. Zangger, Y. Mescam, J. Frey, O. Martin, J. Nicolas, and R. Appel. *Artificial Intelligence Methods and Tools for Systems Biology*, chapter Cooperative metaheuristics for exploring proteomic data. Computational Biology vol 5. Springer, 2004.

[96] R. Gras and J. Nicolas. Forest, a browser for huge dna sequences. In *GIW'96 Genome Informatics 7*, pages 147–156, Tokyo, Japan, dec. 1996.

[97] Robin Gras. *Un outil interactif de recherche de motifs dans les grandes séquences génétiques fondé sur l'arbre des suffixes.* PhD thesis, Université de Rennes I, dec. 1997.

[98] Richard Groult, Martine Léonard, and Laurent Mouchard. A linear algorithm for the detection of evolutive tandem repeats. *Journal of Automata, Languages and Combinatorics*, 10(5/6) :671–685, 2005.

[99] Dan Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK, 1997.

[100] Michael Hackenberg, Christopher Previti, Pedro L. Luque-Escamilla, Pedro Carpena, Jose Martinez-Aroza, and Jose L. Oliver. Cpgcluster : A distance-based algorithm for cpg-island detection. *BMC Bioinformatics*, 7 :446+, October 2006.

[101] Niina Haiminen and Heikki Mannila. Discovering isochores by least-squares optimal segmentation. *Gene*, 394(1-2) :53–60, February 2007.

[102] M. A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, Reading, MA, USA, 1978.

[103] Carsten Helgesen and Peter R. Sibbald. PALM - A pattern language for molecular biology. In L. Hunter, David B. Searls, and J. Shavlik, editors, *In Proceedings of the First International Conference on Intelligent Systems for Molecular Biology(ISMB-93)*, pages 172–180, Menlo Park, CA, 1993. AAAI Press.

[104] David Hernández, Robin Gras, and Ron D. Appel. Model : an efficient strategy for ungapped local multiple alignment. *Computational Biology and Chemistry*, 28(2) :119–128, 2004.

[105] Horsman, M. Katie, Bienvenue, M. Joan, Blasier, R. Kiev, Landers, and P. James. Forensic dna analysis on microfluidic devices : A review. *Journal of Forensic Sciences*, 52(4) :784–799, July 2007.

[106] T. Hubbard, D. Barker, E. Birney, G. Cameron, Y. Chen, L. Clark, T. Cox, J. Cuff, V. Curwen, T. Down, R. Durbin, E. Eyras, J. Gilbert, M. Hammond, L. Huminiecki, A. Kasprzyk, H. Lehvaslaiho, P. Lijnzaad, C. Melsopp, E. Mongin, R. Pettett, M. Pocock, S. Potter, A. Rust, E. Schmidt, S. Searle, G. Slater, J. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik, and M. Clamp. The ensembl genome database project. *Nucl. Acids Res.*, 30(1) :38–41, January 2002.

[107] Lucian Ilie and Silvana Ilie. Multiple spaced seeds for homology search. *Bioinformatics*, 23(22) :2969–2977, 2007.

[108] Lucian Ilie and Wojciech Plandowski. Two-variable word equations. In H. Reichel and S. Tison, editors, *STACS'2000 Annual Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 122–132, 2000.

[109] Lucian Ilie, Sheng Yu, and Kaizhong Zhang. Repetition complexity of words. In *COCOON '02 : Proceedings of the 8th Annual International Conference on Computing and Combinatorics*, pages 320–329, London, UK, 2002. Springer-Verlag.

[110] Costas S. Iliopoulos, James A. M. McHugh, Pierre Peterlongo, Nadia Pisanti, Wojciech Rytter, and Marie-France Sagot. A first approach to finding common motifs with gaps. In *Stringology*, pages 88–97, 2004.

[111] I. Jacquemin and J. Nicolas. Disulfide bonds prediction using inductive logic programming. In *Workshop on Constraint Based Methods for Bioinformatics, WCB*, pages 56–65, Sitges, Spain, 2005.

[112] I. Jacquemin and J. Nicolas. Prédiction de ponts disulfures par langages de contrôle. In *CAP '03, Conférence d'Apprentissage*, pages 56–65, Laval, France, 2005.

[113] Ingrid Jacquemin. *Découverte de motifs relationnels en bioinformatique application à la prédiction des ponts disulfures dans les protéines*. PhD thesis, Université de Rennes I, 2006.

[114] Joxan Jaffar. Minimal and complete word unification. *J. ACM*, 37(1) :47–85, 1990.

[115] S. Jakeway, A. de Mello1, and E. Russell. Miniaturized total analysis systems for biological analysis. *Fresenius' Journal of Analytical Chemistry*, 366(6-7) :525–539, 2000.

[116] R. Jansen, J. D. A. Van Embden, W. Gaastra, and L. M. Schouls. Identification of genes that are associated with dna repeats in prokaryotes. *Mol Microbiol*, 43(6) :1565–1575, 2002.

[117] Svante Janson, Stefano Lonardi, and Wojciech Szpankowski. On average sequence complexity. *Theor. Comput. Sci.*, 326(1-3) :213–227, 2004.

[118] H. J. Jeffrey. Chaos game representation of gene structure. *Nucleic Acids Res*, 18(8) :2163–70, 1990.

[119] A. Joshi, K. Vijay-Shanker, and D. Weir. *Foundational issues in Natural Language processing*, chapter The convergence of midly context-sensitive grammar formalisms, pages 31–81. Sells, P. and Shieber, S. and Wasow, T. Eds, MIT Press, Cambridge, MA, 1991.

[120] J. Jurka. Repeats in genomic dna : mining and meaning. *Curr. Opin. Struct. Biol.*, 8 :333–337, 1998.

[121] J. Jurka, V.V. Kapitonov, A. Pavlicek, P. Klonowski, O. Kohany, and J. Walichiewicz. Repbase update, a database of eukaryotic repetitive elements. *Cytogentic and Genome Research*, 110 :462–467, 2005.

[122] D. Karolchik, R. Baertsch, M. Diekhans, T. S. Furey, A. Hinrichs, Y. T. Lu, K. M. Roskin, M. Schwartz, C. W. Sugnet, D. J. Thomas, R. J. Weber, D. Haussler, and W. J. and Kent. The ucsc genome browser database. *Nucleic Acids Res*, 31(1) :51–54, January 2003.

[123] RM. Karp, RE. Miller, and AL. Rosenberg. Rapid identification of repeated patterns in strings, trees and arrays. In *STOC '72 : Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 125–136, New York, NY, USA, 1972. ACM Press.

[124] E. Kawaguchi and T. Endo. On a method of binary-picture representation and its application to data compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2 :27–35, January 1980.

[125] H. H. Kazazian. Mobile elements : drivers of genome evolution. *Science*, 303(5664) :1626–1632, 2004.

[126] J. Kieffer, E. Yang, G. Nelson, and P. Cosman. Universal lossless compression via multilevel pattern matching. *IEEE Transactions on Information Theory*, 46(4) :1227–1245, 2000.

[127] John C. Kieffer and En-Hui Yang. Grammar-based codes : A new class of universal lossless source codes. *IEEE Transactions on Information Theory*, 46(3) :737–754, 2000.

[128] R. D. King, K. E. Whelan, F. M. Jones, P. G. Reiser, C. H. Bryant, S. H. Muggleton, D. B. Kell, and S. G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971) :247–252, January 2004.

[129] M. Kmita and D. Duboule. Organizing Axes in Time and Space ; 25 Years of Colinear Tinkering. *Science*, 301 :331–334, July 2003.

[130] R. Kolpakov and G. Kucherov. Finding maximal repetitions in a word in linear time. In *Proceedings of the 40th IEEE Annual Symposium on Foundations of Computer Science*, pages 596–604, New York, USA, 1999. IEEE Computer Society Press.

[131] R. Kolpakov and G. Kucherov. On maximal repetitions in words. In G. Ciobanu and Gh. Păun, editors, *Proceedings of the 12th International Symposium on Fundamentals of Computation Theory*, volume 1684, pages 374–385, Iasi, Romania, 1999. Springer-Verlag, Berlin.

[132] R. Kolpakov and G. Kucherov. Finding repeats with fixed gap. In *SPIRE '00 : Proceedings of the 7th International Symposium on String Processing Information Retrieval (SPIRE'00)*, page 162, Washington DC, USA, 2000. IEEE Computer Society.

[133] M. Krzywinski. Circos. *CBHD Newsletter*, 40, June 2005.

[134] R. M. Kuhn, D. Karolchik, A. S. Zweig, H. Trumbower, D. J. Thomas, A. Thakkapallayil, C. W. Sugnet, M. Stanke, K. E. Smith, A. Siepel, K. R. Rosenbloom, B. Rhead, B. J. Raney, A. Pohl, J. S. Pedersen, F. Hsu, A. S. Hinrichs, R. A. Harte, M. Diekhans, H. Clawson, G. Bejerano, G. P. Barber, R. Baertsch, D. Haussler, and W. J. Kent. The ucsc genome browser database : update 2007. *Nucleic Acids Res*, 35(Database issue) :668–673, January 2007.

[135] Stefan Kurtz, J. V. Choudhuri, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich. REPuter : the manifold applications of repeat analysis on a genomic scale. *Nucl. Acids Res.*, 29(22) :4633–4642, 2001.

[136] N. Jesper Larsson and Alistair Moffat. Offline dictionary-based compression. In *Data Compression Conference*, pages 296–305, 1999.

[137] Arnaud Lefebvre, Thierry Lecroq, and Joël Alexandre. An improved algorithm for finding longest repeats with a modified factor oracle. *Journal of Automata, Languages and Combinatorics*, 8(4) :647–657, 2003.

[138] A. Leroux and J. Nicolas. Sdtm, une méthode d'inférence grammaticale pour la découverte de motifs dans des ensembles de protéines. In *CAP 2006 Conf. francophone sur l'apprentissage automatique*, 2006.

[139] Aurélien Leroux. *Inférence grammaticale sur des alphabets ordonnés : Application à la découverte de motifs dans des familles de protéines.* PhD thesis, Université de Rennes I, juin 2005.

[140] Sui-wai Leung, Chris Mellish, and Dave Robertson. Basic gene grammars and GNA-chartparser for lnguage processing of escherchia coli promoter DNA sequences. *Bioinformatics*, 17(3) :226–236, 2001.

[141] Wentian Li, Pedro Bernaola-Galván, Fatameh Haghighi, and Ivo Grosse. Applications of recursive segmentation to the analysis of DNA sequences. *Computers & Chemistry*, 26(5) :491–510, 2002.

[142] Alan Wee-Chung Liew, Yonghui Wu, Hong Yan, and Mengsu Yang. Effective statistical features for coding and non-coding DNA sequence classification for yeast, C. elegans and human. *IJBRA*, 1(2) :181–201, 2005.

[143] Vladimir Lifschitz. Answer set programming and plan generation. *Artif. Intell.*, 138(1-2) :39–54, 2002.

[144] C. Loose, K. Jensen, I. Rigoutsos, and G. Stephanopoulos. A linguistic model for the rational design of antimicrobial peptides. *Nature*, 443 :867–869, oct 2006.

[145] Crochemore M, Iliopoulos CS, Mohamed M, and Sagot MF. Longest repeats with a block of don't cares. In *LATIN 2004 : 6th Latin American Symposium*, pages 271–278, Buenos Aires (Argentina), April 5-8 2004.

[146] Gardiner-Garden M. and Frommer M. Cpg islands in vertebrate genomes. *J Mol Biol.*, 196(2) :261–282, Jul 1987.

[147] Izabela Makalowska, Chiao-Feng Lin, and Wojciech Makalowski. Overlapping genes in vertebrate genomes. *Computational Biology and Chemistry*, 29(1) :1–12, 2005.

[148] Harry Mangalam. tacg - a *grep* for dna. *BMC Bioinformatics*, 3 :8, march 2002.

[149] L. Marsan and M.-F. Sagot. Extracting structured motifs using a suffix tree - Algorithms and application to consensus identification. In S. Minoru and R. Shamir, editors, *Proceedings of the 4th Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 210–219, Tokyo, Japan, 2000. ACM Press.

[150] P. Michalak. Rna world - the dark matter of evolutionary genomics. *Journal of Evolutionary Biology*, 19(6) :1768–1774, 2006.

[151] N. Minc and J.-L. Viovy. Microfluidique et applications biologiques : enjeux et tendances. *Comptes Rendus Physique*, 5(5) :565–575, June 2004.

[152] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, March 1997.

[153] F. J. Mojica, C. Díez-Villaseñor, J. García-Martínez, and E. Soria. Intervening sequences of regularly spaced prokaryotic repeats derive from foreign genetic elements. *J. Mol. Evol.*, 60(2) :174–182, 2005.

[154] Michele Morgante, Alberto Policriti, Nicola Vitacolonna, and Andrea Zuccolo. Structured motifs search. *J. Comp Biol*, 12(8) :1065–1082, 2005.

[155] Jan Mrázek and Shaohua Xie. Pattern locator : a new tool for finding local sequence patterns in genomic dna sequences. *Bioinformatics*, 22(24) :3099–3100, 2006.

[156] M. H. Mucchielli-Giorgi, S. Hazout, and P. Tufféry. Predicting the disulfide bonding state of cysteines using protein descriptors. *Proteins*, 46(3) :243–249, February 2002.

[157] Guy A. Narboni. From prolog iii to prolog iv : The logic of constraint programming revisited. *Constraints*, 4(4) :313–336, 1999.

[158] C. Nevill-Manning. *Inferring Sequential Structure*. PhD thesis, University of Waikato, Hamilton, NZ, 1996.

[159] François Nicolas and Eric Rivals. Hardness results for the center and median string problems under the weighted and unweighted edit distances. *J. Discrete Algorithms*, 3(2-4) :390–415, 2005.

[160] Jacques Nicolas. Grammatical inference as unification. Technical Report 3632, Inria, 1999.

[161] Jacques Nicolas, Patrick Durand, Grégory Ranchy, Sébastien Tempel, and Anne-Sophie Valin. Suffix-tree analyser (stan) : looking for nucleotidic and peptidic patterns in chromosomes. *Bioinformatics*, 21(24) :4408–4410, 2005.

[162] José L. Oliver, Pedro Carpena, Michael Hackenberg, and Pedro Bernaola-Galván. Isofinder : computational prediction of isochores in genome sequences. *Nucleic Acids Research*, 32(Web-Server-Issue) :287–292, 2004.

[163] National Research Council (U.S.). Committee on Frontiers at the Interface of Computing and Biology. *Catalyzing inquiry at the interface of computing and biology*. Washington, D.C. : National Academies Press, 2005.

[164] Bucher P. and Bairoch A. A generalized profile syntax for biomolecular sequences motifs and its function in automatic sequence interpretation. In R. Altman, D. Brutlag, P. Karp, R. Lathrop, and David B. Searls, editors, *Second International Conference on Intelligent Systems for Molecular Biology(ISMB-94)*, pages 53–61, Menlo Park, CA, 1994. AAAI Press.

[165] M.A. Palmer, P. Arzberger, J.E. Cohen, A. Hastings, R.D. Holt, J.L. Morse, D. Sumners, and Z. Luthey-Schulten. Accelerating mathematical-biological linkages. Technical report, Joint National Science Foundation-National Institutes of Health Workshop, National Institutes of Health. Bethesda, Maryland, February 12-13 2003.

[166] Tsyh-Wen Pao and John W. Carr III. A solution of the syntactical induction-inference problem for regular languages. *Comput. Lang.*, 3(1) :53–64, 1978.

[167] Giuseppe Della Penna, Benedetto Intrigila, Enrico Tronci, and Marisa Venturini Zilli. Synchronized regular expressions. *Acta Inf.*, 39(1) :31–70, 2003.

[168] F. C. N. Pereira and D. H. D. Warren. Parsing as deduction. In *Proceedings of 21st Annual Meeting of the Association for Computational Linguistics*, june 1983.

[169] Pierre Peterlongo. *Filtrage de séquences d'ADN pour la recherche de longues répétitions multiples*. PhD thesis, Université de Marne la vallée, 2006.

[170] P. A. Pevzner, M. Y. Borodovsky, and A. A. Mironov. Linguistics of nucleotides sequences I : The significance of deviations from mean statistical characteristics and prediction of the frequencies of occurrence of words. *J. Biomol. Struct. Dynamics*, 6 :1013–1026, 1989.

[171] P. A. Pevzner, M. Y. Borodovsky, and A. A. Mironov. Linguistics of nucleotides sequences II : Stationary words in genetic texts and the zonal structure of DNA. *J. Biomol. Struct. Dynamics*, 6 :1027–1038, 1989.

[172] Pavel A. Pevzner. *Computational molecular biology, An algorithmic approach*. MIT Press, Cambridge, MA, 2000.

[173] Pavel A. Pevzner and Sing-Hoi Sze. Combinatorial approaches to finding subtle signals in dna sequences. In *Proc. of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 269–278. AAAI Press, 2000.

[174] T. Pfeiffer, O. S. Soyer, and S. Bonhoeffer. The evolution of connectivity in metabolic networks. *PLoS Biol*, 3(7), July 2005.

[175] Nadia Pisanti, Alexandra M. Carvalho, Laurent Marsan, and Marie-France Sagot. Risotto : Fast extraction of motifs with mismatches. In *Proc. of the 7th Latin American Symposium on Theoretical Informatics (LATIN'06)*, pages 757–768, Valdivia, Chile, 2006.

[176] Wojciech Plandowski. An efficient algorithm for solving word equations. In Jon M. Kleinberg, editor, *STOC*, pages 467–476. ACM, 2006.

[177] G. Polaillon, L. Vescovo, M. Michaut, and J.-C. Aude. *Mining biological data using pyramids*, pages 397–408. Springer, Heidelberg, 2007.

[178] C. Pourcel, G. Salvignol, and Gilles Vergnaud. CRISPR elements in *Yersinia pestis* acquire new repeats by preferential uptake of bacteriophage DNA, and provide additional tools for evolutionary studies. *Microbiology*, 151 :653–63, 2005.

[179] E. Prieur and T. Lecroq. On-line construction of compact suffix vectors and maximal repeats. In *Actes des Journ´ees Montoises (JM 2006)*, pages 311–322, Rennes, 2006.

[180] P. Quignon, M. Giraud, M. Rimbault, P. Lavigne, S. Tacher, E. Morin, E. Retout, A.-S. Valin, K. Lindblad-Toh, J. Nicolas, and F. Galibert. The dog and rat olfactory receptor repertoires. *Genome Biology*, 6(10) :R83, 2005.

[181] Mathieu Raffinot. On maximal repeats in strings. *Information Processing Letters*, 80(3) :165–169, 2001.

[182] Mathieu Raffinot. On maximal repeats in strings. *Information Processing Letters*, 80(3) :165–169, November 2001.

[183] Arcot Rajasekar. Applications in constraint logic programming with strings. In *Principles and Practice of Constraint Programming*, pages 109–122, 1994.

[184] Arcot Rajasekar. String-oriented databases. In *SPIRE '99 : Proceedings of the String Processing and Information Retrieval Symposium & International Workshop on Groupware*, pages 158–167, Washington, DC, USA, 1999. IEEE Computer Society.

[185] J. Raymond and D. Segrè. The effect of oxygen on biochemical networks and the evolution of complex life. *Science*, 311(5768) :1764–1767, March 2006.

[186] Philip G. K. Reiser, Ross D. King, Douglas B. Kell, Stephen Muggleton, Christopher H. Bryant, and Stephen G. Oliver. Developing a logical model of yeast metabolism. *Electron. Trans. Artif. Intell.*, 5(B) :223–244, 2001.

[187] D. Reyes, D. Iossifidis, P.-A. Auroux, and A. Manz. Micro total analysis systems. 1. introduction, theory, and technology. *Analytic Chemistry*, 74 :2623–2636, 2002.

[188] Christian M. Ruitberg, Dennis J. Reeder, and John M. Butler. Strbase : a short tandem repeat dna database for the human identity testing community. *Nucleic Acids Research*, 29(1) :320–322, 2001.

[189] Marie-France Sagot and Eugene W. Myers. Identifying satellites and periodic repetitions in biological sequences. *Journal of Computational Biology*, 5(3) :539–554, 1998.

[190] Sakakibara. Learning context-free grammars from structural data in polynomial time. *TCS : Theoretical Computer Science*, 76, 1990.

[191] Yasubumi Sakakibara. Grammatical inference in bioinformatics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7) :1051–1062, 2005.

[192] K. Salomaa. Patterns. In Gh. Paun C. Martin-Vide, V. Mitrana, editor, *Formal Languages and Applications : Studies in Fuzziness and Soft Computing 148*, pages 367–379, Berlin, 2004. Springer.

[193] D. Schausi, V. Vallet-Erdtmann, C. Tiffoche, G.Tilly, M. Guerrois, B. Jégou, ML Thieulant, J. Nicolas, and S. Tempel. Regulation of an intronic promoter of rat estrogen receptor alpha gene. targeting of promoter to the pituitary in transgenic mice, 2004. Conférence Découverte des génomes et expression des gènes, Paris mai 2003 (Poster).

[194] S. Schwartz, Z. Zhang, K. A. Frazer, A. Smit, C. Riemer, J. Bouck, R. Gibbs, R. Hardison, and W. Miller. Pipmaker-a web server for aligning two genomic dna sequences. *Genome Res*, 10(4) :577–586, 2000.

[195] David B. Searls. Representing genetic information with formal grammars. In *Proceedings of the Seventh National Conference on Artificial Intelligence, AAAI-88*, volume 2, pages 386–391, San Mateo, CA., August 1988. Morgan Kauffman Publishers, Inc.

[196] David B. Searls. Investigating the linguistics of DNA with definite clause grammars. In Ewing L. Lusk and Russ A. Overbeek, editors, *Logic Programming : Proceedings of the North American Conference*, volume 1 of *Logic programming series*, pages 189–208. MIT Press, Cambridge, Mass., 1989.

[197] David B. Searls. The linguistics of DNA. *American Scientist*, 80 :579–591, 1992.

[198] David B. Searls. The computational linguistics of biological sequences. In Larry Hunter, editor, *Artificial Intelligence and Molecular Biology*, chapter 2, pages 47–120. AAAI Press, 1993.

[199] David B. Searls. Genome informatics. *IEEE Computers in Medicine and Biology*, 12(4) :124, 1994.

[200] David B. Searls. Formal grammars for intermolecular structure. In *First international symposium on intelligence in neural and biological systems. INBS'95*, pages 30–37, Los Alamitos, CA., 1995. IEEE Computer Society Press.

[201] David B. Searls. String variable grammar : A logic grammar formalism for the biological language of DNA. *Journal of Logic Programming*, 24(1 & 2) :73–102, July/August 1995.

[202] David. B. Searls. Reading the book of life. *Bioinformatics*, 17(7) :579–580, July 2001.

[203] David. B. Searls. The language of genes. *Nature*, 420 :211–217, November 2002.

[204] David B. Searls and Shang Dong. A syntactic pattern recognition system for DNA sequences. In C. R. Cantor, H. A. Lim, J. Fickett, and R. J. Robbins, editors, *Proceedings 2nd International Conference on Bioinformatics, Supercomputing, and Complex Genome Analysis*, pages 89–101. World Scientific, 1993.

[205] F. Servant, C. Bru, S. Carrère, E. Courcelle, J. Gouzy, D. Peyruc, and D. Kahn. Prodom : automated clustering of homologous domains. *Brief Bioinform*, 3(3) :246–251, September 2002.

[206] A. N. Skourikhine and T. Burr. Linguistic analysis of the nucleoprotein gene of influenza a virus. In *BIBE '00 : Proceedings of the 1st IEEE International Symposium on Bioinformatics and Biomedical Engineering*, pages 193–199, Washington, DC, USA, 2000. IEEE Computer Society.

[207] Rhazes Spell, Rachael Brady, and Fred Dietrich. Bard : A visualization tool for biological sequence analysis. In *INFOVIS*. IEEE Computer Society, 2003.

[208] D. Stanojevic, S. Small, and M. Levine. Regulation of a Segmentation Stripe by Overlapping Activators and Repressors in the Drosophila Embryo. *Science*, 254 :1385–1387, November 1991.

[209] P. Stothard and D. S. Wishart. Circular genome visualization and exploration using cgview. *Bioinformatics*, 21(4) :537–539, February 2005.

[210] Jens Stoye and Dan Gusfield. Simple and flexible detection of contiguous repeats using a suffix tree. *Theor. Comput. Sci.*, 270(1-2) :843–856, 2002.

[211] Yanni Sun and Jeremy Buhler. Designing multiple simultaneous seeds for dna similarity search. *Journal of Computational Biology*, 12(6) :847–861, 2005.

[212] Tanatsugu. A grammatical inference for context-free languages based on self-embedding. *BINFCYB : Bulletin of Informatics and Cybernetics*, 22, 1987.

[213] W. R. Taylor. The classification of amino acid conservation. *J Theor Biol*, 119(2) :205–218, March 1986.

[214] S. Tempel, M. Giraud, D. Lavenier, I.-C. Lerman, A.-S. Valin, I. Couee, A. El Amrani, and J. Nicolas. Domain organization within repeated dna sequences : application to the study of a family of transposable elements. *Bioinformatics*, 22(16) :1948 – 1954, 2006.

[215] S. Tempel, J. Nicolas, I. Couee, and A. El Amrani. The combinatorics of helitron termini in a. thaliana genome revealed strongly structured superfamilies. In *1st international Conference Genomic Impact of Aukariotic Transposable Elements*, Pacific Grove, CA, USA, 2006. The Asilomar Conference Center.

[216] Sébastien Tempel. *Dynamique des hélitrons dans le génome d'Arabidopsis thaliana : développement de nouvelles stratégies dans l'analyse des éléments transposables.* PhD thesis, Université de Rennes I, juin 2007.

[217] Sébastien Tempel, Jacques Nicolas, Abdelhak El Amrani, and Ivan Couée. Model-based identification of helitrons results in a new classification of their families in arabidopsis thaliana. *Gene*, 403(1-2) :1299–1305, 15 nov 2007.

[218] Nam Tran. *Reasoning and hypothesizing about signaling networks.* PhD thesis, Arizona state University, dec. 2006.

[219] Olgierd Unold and Lukasz Ciel. Learning context-free grammars from partially structured examples : Juxtaposition of gcs with tbl. In *HIS '07 : Proceedings of the 7th International Conference on Hybrid Intelligent Systems (HIS 2007)*, pages 348–352, Washington, DC, USA, 2007. IEEE Computer Society.

[220] P. Veber. *Modélisation grande échelle de réseaux biologiques : vérification par contraintes booléennes de la cohérence des données*. PhD thesis, Université de Rennes I, dec. 2007.

[221] Philippe Veber, Sébastien Tempel, Rumen Andonov, Dominique Lavenier, and Jacques Nicolas. Détection de domaines dans des séquences génomiques : un problème de couverture optimale. In *Francoro V / Roadef 2007*, pages 433–434. Presses Universitaires de Grenoble, 2007. Résumé étendu.

[222] E. Verpoorte. Microfluidic chips for clinical and forensic analysis. *Electrophoresis*, 23(5) :677–712, Mar 2002.

[223] D. S. Wishart, D. Tzur, C. Knox, R. Eisner, A. C. Guo, N. Young, D. Cheng, K. Jewell, D. Arndt, S. Sawhney, C. Fung, L. Nikolai, M. Lewis, M. A. Coutouly, I. Forsythe, P. Tang, S. Shrivastava, K. Jeroncic, P. Stothard, G. Amegbey, D. Block, D. D. Hau, J. Wagner, J. Miniaci, M. Clements, M. Gebremedhin, N. Guo, Y. Zhang, G. E. Duggan, G. D. Macinnis, A. M. Weljie, R. Dowlatabadi, F. Bamforth, D. Clive, R. Greiner, L. Li, T. Marrie, B. D. Sykes, H. J. Vogel, and L. Querengesser. Hmdb : the human metabolome database. *Nucleic Acids Res*, 35(Database issue), January 2007.

[224] En-Hui Yang and John C. Kieffer. Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform - part one : Without context models. *IEEE Transactions on Information Theory*, 46(3) :755–777, 2000.

[225] Takashi Yokomori. Learning non-deterministic finite automata from queries and counterexamples. In *Machine Intelligence 13*, pages 169–189, 1994.

[226] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5) :530–536, 1978.

"Referme ton livre. Pense librement et regarde librement le ciel et la terre."
Omar Khayyâm